ABSTRACT

Title of dissertation: DISCRIMINATIVE FEATURE LEARNING WITH APPLICATION TO FINE-GRAINED RECOGNITION

Yaming Wang, Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Department of Electrical and Computer Engineering

For various computer vision tasks, finding suitable feature representations is fundamental. Fine-grained recognition, distinguishing sub-categories under the same super-category (*e.g.*, bird species, car makes and models *etc.*), serves as a good task to study discriminative feature learning for visual recognition task. The main reason is that the inter-class variations between fine-grained categories are very subtle and even smaller than intra-class variations caused by pose or deformation.

This thesis focuses on tasks mostly related to fine-grained categories. After briefly discussing our earlier attempt to capture subtle visual differences using sparse/low-rank analysis, the main part of the thesis reflects the trends in the past a few years as deep learning prevails.

In the first part of the thesis, we address the problem of fine-grained recognition via a patch-based framework built upon Convolutional Neural Network (CNN) features. We introduce triplets of patches with two geometric constraints to improve the accuracy of patch localization, and automatically mine discriminative

geometrically-constrained triplets for recognition.

In the second part we begin to learn discriminative features in an end-to-end fashion. We propose a supervised feature learning approach, Label Consistent Neural Network, which enforces direct supervision in late hidden layers. We associate each neuron in a hidden layer with a particular class and encourage it to be activated for input signals from the same class by introducing a label consistency regularization. This label consistency constraint makes the features more discriminative and tends to faster convergence.

The third part proposes a more sophisticated and effective end-to-end network specifically designed for fine-grained recognition, which learns discriminative patches within a CNN. We show that patch-level learning capability of CNN can be enhanced by learning a bank of convolutional filters that capture class-specific discriminative patches without extra part or bounding box annotations. Such a filter bank is well structured, properly initialized and discriminatively learned through a novel asymmetric multi-stream architecture with convolutional filter supervision and a non-random layer initialization.

In the last part we goes beyond obtaining category labels and study the problem of continuous 3D pose estimation for fine-grained object categories. We augment three existing popular fine-grained recognition datasets by annotating each instance in the image with corresponding fine-grained 3D shape and ground-truth 3D pose. We cast the problem into a detection framework based on Faster/Mask R-CNN. To utilize the 3D information, we also introduce a novel 3D representation, named as location field, that is effective for representing 3D shapes.

Discriminative Feature Learning with Application to Fine-grained Recognition

by

Yaming Wang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor David W. Jacobs
Professor Joseph F. JaJa
Professor Min Wu

Dedication

To my dearest parents.

Acknowledgments

First of all, I would like to thank my advisor, Prof. Larry Davis. I would never forget his kindness five years ago, when he offered me, without any hesitation, the precious opportunity into the world of computer vision. During the past five years, He is always knowledeable and kindly approachable, giving me insightful advice whenever I made research or career decisions. He provided me opportunities to work on various exciting problems as well as allowing me to explore my interesting area freely. It is his wise guidance and generous support that make me flourish and mature as a qualified computer vision professional.

I would like to thank Prof. Rama Chellappa, Prof. David Jacobs, Prof. Joseph JaJa and Prof. Min Wu for kindly serving as my dissertation committee and for kind advice during dissertation preparation. In addition, I would like to thank Prof. Min Wu for her advice at the beginning of my Ph.D. years, who encouraged me to be brave and face the real life.

I would like to thank my collaborators/mentors: Vlad Morariu, Zhuolin Jiang, Feng Zhou, Jonghyun Choi, Yi Yang, Xiao Tan and Xiao Liu. Working with them is a pleasure and I appreciate their guidance and advice as well as the opportunities they offered. I would also like to thank my lab-mates who have worked closely with me: Sohil Shah, Ang Li, Mingfei Gao and Yen-Liang Lin. I have learned a lot from them and appreciate their help and inspirations. I would like to thank all my other colleagues in the computer vision lab who make my graduate study colorful: Fan Yang, Yangmuzi Zhang, Joe Ng, Guangxiao Zhang, Bharat Singh, Xintong Han,

Xiyang Dai, Zhe Wu, Zuxuan Wu, just to name a few.

I would like to thank my friends at the University of Maryland: Hua He, Ren Mao, Wentao Luan, Zhihao Li, Hao Zhou, Jianwei Xie, Bingxin Yang and Shuo Sun. Life with them during the past years is a pleasure and I thank them for their kind and generous help in my everyday life. I would also like to thank some old friends: Jun Gao, Rui Ye, Qing Sun, Jie Ding, Keyi Wang, Haiyan Deng and Fanbo Ji, for their understanding and support during the past years.

Finally, I owe my deepest thanks to my parents for their selfless love and conditionless support during all these years. They always stand behind me as a strong backup and helped me through some of the most challenging times during my Ph.D. era. At completion time, I have had a much deeper understanding of some of the life lessons they taught me long ago, which is my life-time treasure.

Table of Contents

Dedication ii

Acknowledgements iii

List of Tables viii

List of Figures x

1 Introduction 1
- 1.1 Background and Motivation . . . 1
- 1.2 Approaches . . . 3
  - 1.2.1 Publications . . . 8

2 Unsupervised Feature Extraction Inspired by Latent Low-Rank Representation 9
- 2.1 Background and Motivation . . . 9
- 2.2 Analysis of LatLRR's Feature Extraction . . . 11
  - 2.2.1 Preliminary . . . 12
  - 2.2.2 Empirical Analysis . . . 14
  - 2.2.3 Theoretical Analysis . . . 18
- 2.3 Unsupervised Feature Extraction Inspired by Latent Low-rank Representation . . . 26
- 2.4 Experiments . . . 29

3 Mining Discriminative Triplets of Patches for Fine-Grained Classification 33
- 3.1 Motivation . . . 33
- 3.2 Triplets of Patches with Geometric Constraints . . . 35
  - 3.2.1 Order Constraint and Shape Constraint . . . 35
  - 3.2.2 Triplet Detector . . . 38
- 3.3 Discriminative Triplets for Fine-Grained Classification . . . 40
  - 3.3.1 Triplet Initialization . . . 41
  - 3.3.2 Discriminative Triplets Mining by Entropy Scores . . . 44
  - 3.3.3 Mid-Level Image Representations for Classification . . . 45

3.4 Experiments . . . 46
3.4.1 Triplet Localization with Geometric Constraints . . . 46
3.4.2 14-Class BMVC Cars Dataset Results . . . 48
3.4.3 196-Class Stanford Cars Dataset Results . . . 52
3.4.4 100-Class FGVC-Aircraft Dataset Results . . . 57

4 Learning Discriminative Features via Label Consistent Neural Network 58
4.1 Background and Motivation . . . 58
4.2 Feature Learning via Supervised Deep Learning . . . 60
4.3 Label Consistent Neural Network (LCNN) . . . 61
4.3.1 Motivation . . . 61
4.3.2 Formulation . . . 62
4.3.3 Network Training . . . 66
4.4 Experiments . . . 66
4.4.1 Action Recognition . . . 67
4.4.2 Object Recognition . . . 73

5 Learning a Discriminative Filter Bank within a CNN for Fine-grained Recognition 79
5.1 Background and Motivation . . . 79
5.2 Related Work . . . 82
5.3 Learning Discriminative Patch Detectors as a Bank of Convolutional Filters . . . 84
5.3.1 Asymmetric Two-stream Architecture . . . 86
5.3.2 Convolutional Filter Supervision . . . 87
5.3.3 Layer Initialization . . . 89
5.3.4 Extension: Multiple Scales . . . 90
5.4 Experiments . . . 91
5.4.1 Implementation Details . . . 91
5.4.2 Results . . . 92
5.4.3 Ablation Studies . . . 95
5.4.4 Visualization and Analysis . . . 97
5.4.4.1 Stanford Cars . . . 98
5.4.4.2 CUB-200-2011 . . . 101

6 Continuous 3D Pose Estimation for Fine-grained Objects 102
6.1 Motivation . . . 102
6.2 Related Work . . . 105
6.2.1 Fine-Grained Recognition . . . 105
6.2.2 Monocular 3D Pose Estimation . . . 106
6.2.3 Monocular 3D Pose Estimation Dataset . . . 109
6.2.4 Joint Object Recognition and 3D Pose Estimation . . . 110
6.2.5 3D Representation . . . 111
6.3 Dataset . . . 111
6.3.1 3D models . . . 111

6.3.2 Camera model . . . 112
6.3.3 3D Annotation . . . 113
6.4 Continuous 3D Pose Estimation for Fine-Grained Objects . . . 114
6.4.1 Baseline Framework . . . 115
6.4.2 Improve Pose Estimation via 3D Location Field . . . 118
6.4.3 Joint Pose Estimation and Fine-grained Recognition . . . 119
6.5 Baseline Experiments . . . 120
6.5.1 Evaluation Metrics . . . 120
6.5.2 Experimental Settings . . . 122
6.5.3 Results and Analysis . . . 123
6.6 Extended Experiments . . . 125

7 Conclusion 127

Bibliography 129

List of Tables

2.1 The simplification of Algorithm 1 by focusing on the operations of the singular values. . . . 20
2.2 Our feature extraction procedure. . . . 28
2.3 Classification accuracies (%, averaged over 20 runs) on Extended Yale Database B. For fair comparison, the results related to raw data and LatLRR are cited from [1], who chose the dimension 317D to obtain the best result within the range of 400D. . . . 31
2.4 Classification accuracies (%, averaged over 20 runs) on CMU PIE face databases. . . . 31
2.5 Results of efficiency test (averaged over 20 runs) on Extended Yale Database B. . . . 32

3.1 Triplets localization test result on FG3DCar dataset. The localization accuracy and relative improvement over baseline (Appearance Only method) are demonstrated. . . . 48
3.2 Results on BMVC-14 dataset. . . . 50
3.3 Results on Cars-196 dataset. Items with "*" indicate that no extra annotations/data are involved. . . . 56
3.4 Results on FGVC-Aircraft dataset. . . . 56

4.1 Classification performances with different two-stream CNN approaches on the UCF101 dataset (split-1). The results of [2], [3] and [4] are copied from their original paper. The VGGNet-16* result is obtained by running the original model trained and shared by [2], while VGGNet-16** is the reproduced result by using the same parameters and initial model provided by [2]. . . . 70
4.2 Recognition performance comparisons with state-of-the-art approaches on the UCF101 dataset. . . . 70
4.3 Mean Average Precision performance on the THUMOS15 validation set. The results of [2], [3] and [5] are copied from [2]. VGGNet-16* is the result of using the softmax loss only to train the network. Our result 62.6% mAP is also better than 54.7% using method in [6], which is reported in [7]. . . . 74

4.4 Test error rates from different approaches on the CIFAR-10 dataset. . 75
4.5 Recognition Performances using different approaches on the ImageNet 2012 Validation set. The result of GoogLeNet [5] is copied from original paper while GoogLeNet* [5] is the reproduced result by using the same parameters provided in [5]. The result of LCNN-2 is obtained by training the network from scratch under the same training condition as GoogLeNet* [5]. 76
4.6 Comparisons of LCNN with other approaches on the Caltech101 dataset. The results of LCNN-2* are obtained by using VGGNet-16 as the underlying architecture while the results of LCNN-1 and LCNN-2 are based on AlexNet. 78

5.1 Comparison of our approach (DFL-CNN) to recent results on CUB-200-2011, **without** extra annotations (if not specified). For the fine-tuned (FT) baselines, we cite the best previously reported result if it is better than our implementation. The black-bold number represents the best previous result. 94
5.2 Comparison of our approach (DFL-CNN) to recent results on Stanford Cars **without** extra annotations (if not specified). 94
5.3 Comparison of our approach (DFL-CNN) to recent results on FGVC-Aircraft **without** extra annotation (if not specified). 94
5.4 Contribution of the streams *at test time* on CUB-200-2011. Note that at training time a *full* DFL-CNN model is trained, but the prediction only uses certain stream(s). 95
5.5 Effect of Global Max Pooling (GMP) vs. Global Average Pooling (GAP) on CUB-200-2011. 95
5.6 Effect of intermediate supervision of DFL-CNN *at training time*, evaluated on CUB-200-2011. 96
5.7 Effect of BBox evaluated on CUB-200-2011. 96

6.1 Comparison of our dataset with some of the other 3D datasets. 102
6.2 Train / Test Split of the Datasets. 122
6.3 Baseline results of 3D pose estimation. 123
6.4 Extended experimental results on Stanford Cars 3D 125

List of Figures

2.1 The optimization ordering that yields the best performance (red) down-weights the most significant principal components (low i), while the other ordering (black) gives them more weight. Top: the singular values of transformation matrix L. Bottom: the singular values of data matrix X and transformed data LX. The latter is displayed in a semi-log graph due to the large variations in scale. 15

2.2 An example principal component (left) and the component corresponding to relatively small σ_{Xi} (right). On the left is the u_{Xi} corresponding to the largest σ_{Xi}, and on the right is the one corresponding to the largest $l_{1i}\sigma_{Xi}$ after reweighting. 17

2.3 l as a function of α given by (2.21). The definition of α is given in (2.15). For small σ_X, α approaches 0 and l decreases very slowly from 1/2; for very large σ_X, α approaches 1 and l drops very quickly. Parameter settings: $k_0 = 6$. 24

2.4 The singular values of our theoretical approximation (red) compared to those achieved by ALM in practice (blue), where l_i denotes the i^{th} singular value of L. Our theoretical approximation very closely models the behavior of ALM. Parameter settings: $\mu_0 = 10^{-6}, \rho = 5$, $\epsilon = 10^{-4}$ (practice); $k_0 = 6$ (theory). 25

2.5 The singular values of L learnt by our method, where l_i denotes the i^{th} singular value. By comparing with Figure 2.4, we can see that our method has the reweighting effect similar to LatLRR. 31

2.6 The visualization of the decomposition by our method. In each group of the same individual, the original data X (left) is decomposed into a principal part XZ^* (middle) and a detailed part L^*X (right). . . . 32

3.1 Visualization of the order constraint. Top: Patch A, B and C are arranged in clockwise order. The direction of $\overrightarrow{AB} \times \overrightarrow{AC}$ points into the paper, and $G_{ABC} = 1$. Bottom: A', B' and C' are arranged in counterclockwise order. The direction of the cross product points out of the paper, and $G_{A'B'C'} = -1$. 37

3.2 Visualization of the shape constraint. The constraint compares the three angles of two triplets. 37

3.3 Summary of our discriminative triplet mining framework. Candidate triplets are initialized from sets of neighboring images and selected by how discriminative they are across the training set. The mid-level representation consists of the maximum responses of the selected triplets with geometric constraints, which is fed into a linear SVM for classification. 40
3.4 Visualization of the triplet initialization stage. **(a)** A seed image from class c_0 is used to construct its nearest-neighbor set including itself. **(b)** Discriminative score map is generated from the stack of neighboring images. **(c)** Patch locations with top discriminative scores are selected by non-maximum suppression. **(d)** Images from positive class (class c_0) are selected to generate triplets. **(e)** Triplets are generated from positive samples at locations proposed in (c). 42
3.5 Some examples of a set of neighboring images. The query image is highlighted with a red box. Since fine-grained objects share similar overall appearance, the neighborhood consists of samples from different classes with the same pose. 42
3.6 Visualization of the effects of the two geometric constraints. Red boxes are incorrectly localized patches. The order constraint roughly checks the geometric arrangement of three patches and can eliminate incidental false detections which happen to have high appearance score; the shape constraints enforce finer adjustments on patch locations than the order constraint. 45
3.7 Visualization of the most discriminative triplet (measured by Eq. (3.12)) for each class in BMVC-14 proposed by our method. The triplets accurately capture the subtle discriminative information of each class, which is highly consistent with human perception. For instance, for the **first image in the first row**, the triplet captures the curvy nature of Volkswagen Beetle such as rounded hood; for the **first image in the second row**, the triplet focuses on the rear cargo of the pick-up truck, since Ford F-Series is the only pick-up in the dataset; for the **last image in the first row**, the triplet highlights the frontal face of Jeep Wrangler. 51
3.8 Classification accuracy with respect to the number of triplets per class. 52
3.9 (a) Visualization of the most discriminative triplets of two example classes in Cars-196. (b)(c) The averaged image-level BoT descriptor across all test samples in the corresponding class. Each dimension in the BoT is generated by the response of a mined triplet. The color bars are used to describe the dimensions correponding to the responses of triplets from different classes. 54

4.1 An example of the structure of LCNN. The label consistency module is added to the first late hidden layer, which is a fully-connected layer fc_l. Its representation $\mathbf{x}^l$ is transformed to be $\mathbf{A}^{(l)}\mathbf{x}^l$, which is the output of the transformed representation layer $\text{fc}_{l+0.5}$. Note that the applicability of the proposed label consistency module is not limited to fully-connected layers. 63
4.2 Class 4 (BabyCrawling) and class 10 (BenchPress) samples from the UCF101 action dataset. 67
4.3 Examples of direction supervision in the late hidden layers including (a) fc_7 layer in the CNN architectures including VGG [8] and AlexNet [9]; (b) CCCP5 layer in the Network-in-Network [10];(c) loss_1/fc, loss_2/fc and $\text{Pool}_5/7\times7\text{S}_1$ in the GoogLeNet [5]. The symbol of three dots denotes other layers in the network. 68
4.4 Examples of learned representations from layers fc_6, fc_7 and $\text{fc}_{7.5}$ using LCNN and VGGNet-16. Each curve indicates an average of representations for different testing videos from the same class in the UCF101 dataset. The first two rows correspond to class 4 (Baby Crawling, 35 videos) while the third and fourth rows correspond to class 10 (Bench Press, 48 videos). The curves in every two rows correspond to the spatial net (denoted as 'S') and temporal net (denoted as 'T') in our two-stream framework for action recognition. (a) fc_6 representations using VGGNet-16; (b) Histograms (with 100 bins) for representations from (a); (c) fc_6 representations using LCNN; (d) Histograms for representations from (c); (e) fc_7 representations using VGGNet-16; (f) Histograms for representations from (e); (g) fc_7 representations using LCNN; (h) Histograms for representations from (g); (i) $\text{fc}_{7.5}$ representations (i.e. transformed fc_7 representations) using LCNN. The entropy values for representations from (a)(c)(e)(g) are computed as: (11.32, 11.42, 11.02, 10.75), (11.2, 11.14, 10.81, 10.34), (11.08, 11.35, 10.67, 10.17), (11.02, 10.72, 10.55, 9.37). LCNN can generate lower-entropy representations for each class compared to VGGNet-16. The figure is best viewed in color and 600% zoom in. 69
4.5 Training and testing error comparisons (based on spatial net) between LCNN-2 and VGGNet-16 on the UCF101 dataset. (a) Training error comparison; (b) Testing error comparison. 71
4.6 Effects of parameter selection of k-NN neighborhood size k on the classification accuracy performances on the UCF101 dataset. 72

5.1 The motivation of our approach is to regard a $C \times 1 \times 1$ vector in a feature map as the representation of a small patch and a 1×1 convolutional filter as a discriminative patch detector. A discriminative patch can be discovered by convolving the feature map with the 1×1 filter and performing Global Max Pooling (GMP) over the response map. The full architecture is illustrated in Figure 5.2. 81

5.2 Overview of our framework, which consists of a) an asymmetric two-stream architecture to learn both the discriminative patches and global features, b) supervision imposed to learn discriminative patch detectors and c) non-random layer initialization. For simplicity, except GMP, all pooling and ReLU layers between convolutional layers are not displayed. 86
5.3 The illustration of our convolutional filter supervision. The filters in `conv6` are grouped into M groups, where M is the number of classes. The maximum responses in group i are averaged into a single score indicating the effect of the discriminative patches in Class i. The pooled vector is fed into a softmax loss layer to encourage discriminative patch learning. 87
5.4 The visualization of top patches in Stanford Cars. We remap the spatial location of the highest activation in a feature map back to the patch in the original image. The results are highly consistent with human perception, and cover diverse regions such as head light (2^{nd} **column**), air intake (3^{th} **column**), frontal face (4^{th} **column**) and the black side stripe (**last column**). 96
5.5 Sample visualization of all ten filter activations learned for one class (Class 102) by upsampling the `conv6` feature maps to image resolution, similar to [11]. The activations are disriminatively concentrated and cover diverse regions. Better viewed at 600%. 97
5.6 The `pool6` features averaged over all test samples from Class 10, 101 and 151 in Stanford Cars. The dash lines indicate the range of values given by the discriminative patch detectors belonging to the class. The representations peak at the corresponding class. 97
5.7 Visualization of a failure case, where the filter activates on commonly appeared licence plates. 98
5.8 The visualization of patches in CUB-200-2011. We accurately localize discriminative patches without part annotations, such as the bright texture (**first image**), the color spot (**second image**), the webbing and beak (**third** and **forth** image). 98
5.9 The averaged `pool6` features over all test samples from Class 101 in CUB-200-2011, peaky at corresponding dimensions. 99

5.10 Visualization of the energy distribution of `conv4_3` feature map before and after training for Stanford Cars. We remap each spatial location in the feature map back to the patch in the original image. After training in our approach, the energy distribution becomes more discriminative. For example, in the 1st **column**, the high energy region shifts from the wheels to discriminative regions like the frontal face and the top of the vehicle; in the 2nd **column**, after training the energy over the brick patterns is reduced; in the 3rd **column**, the person no longer lies in high energy region after training; in the 7th **column**, before training the energy is focused mostly at the air grill, and training adds the discriminative fog light into the high energy region. More examples are interpretated in Section 5.4.4.1. 100
5.11 The energy distributions of `conv4_3` feature maps before and after training in CUB-200-2011. After training, in the left example, the high energy region at the background branches is greatly shrinked and the energy is concentrated at the discriminative color spot; in the right example, more energy is distributed to the distinctive black-and-white wing and tail of the species. 101

6.1 Overview of our annotation interface. 114
6.2 Pose Estimation Framework Diagram 117
6.3 Sample image and its corresponding 3D location fields 119
6.4 Left: Network architecture of using 3D location field to help pose. Right: Network architecture for joint fine-grained recognition and pose estimation. 120
6.5 Plot of Acc_{th} *w.r.t.* threshold. 123
6.6 Visualizations of predicted poses for test samples. For each dataset, we show five examples of successful predictions and two of the failure cases, separated by the solid black line in the figure. 124

Chapter 1: Introduction

1.1 Background and Motivation

Visual Feature Learning Learning suitable feature representations lies at the core of various computer vision tasks. In terms of image recognition, the fundamental challenge is to make the features discriminative. Recent years have seen dramatic changes in feature learing for image recognition. Earlier multi-stage frameworks extract hand-crafted local feature descriptors from raw image, and encode them into an image-level feature vector as the input to a classifier. After the breakthrough of Convolutional Neural Network (CNN) in 2012, visual recognition has gradually shifted from multi-stage frameworks to the end-to-end deep learning frameworks. In these frameworks, the low-level feature extraction, mid-level feature encoding and high-level classification are integrated into a single neural network and trained end-to-end using back-propogation.

Fine-grained Recognition At the same time around 2012, a new type of recognition task, fine-grained recognition, was proposed, which aims to distinguish large numbers of sub-categories from the same super-category (*e.g.*bird species, dog breeds, makes and models of cars and aircrafts). It serves as a good task to study discriminative feature learning for two reasons. The main reason is that the inter-class

variations between fine-grained categories are very subtle and even much smaller than intra-class variations caused by pose and deformation. The second reason is that the number of training samples per class is small while the number of classes is large, since fine-grained category labels need expert knowledge that are thus more expensive to obtain. The subtle visual differences and the limited training data together require higher learning capability from the recognition framework.

Feature Learning for Fine-grained Recognition There have been extensive efforts on discriminative feature learning for fine-grained recognition during the past five years, probably due to the fact that classical CNN architectures do not work very well on the task. Researchers soon found that *(i)* directly fine-tuning earlier CNN architectures such as AlexNet did no better than sophisticated feature encoding with hand-crafted features; *(ii)* distinguishing subtle differences depends heavily on discriminative highly-localized regions such as semantic part or patches. Since then, research on fine-grained recognition can be divided into four eras. The works during the first era were multi-stage frameworks depending heavily on semantic part annotations; in the second era, researchers tried to eliminate the expensive part annotation and developed various multi-stage frameworks built upon off-the-shelf CNN feature representations of image regions; the end-to-end CNN-based frameworks marks the characteristics of the third era; recently, researchers have been trying to obtain information more than just the category label. The performance has improved a lot during this process.

Overall Motivation To summarize, the underlying motivation behind fine-grained recognition research is essentially *enhancing the mid-level learning capa-*

bility of the recognition system. For multi-stage frameworks, research was focused on generating discriminative mid-level features from low-level local descriptors; for CNN-based approaches, research was focused on designing proper network structures to improve the learning capabilities of intermediate convolutional layers. In these ways, the prior that "the problem depends on localized discriminative regions" is injected into the recognition system, as the approaches of the thesis will show below.

1.2 Approaches

This thesis focuses on tasks mostly related to fine-grained categories. After briefly discussing our earlier attempt to capture subtle visual differences in Chapter 2, the main part reflects the trends in the past a few years as dicussed above.

Our earlier work in **Chapter 2** attempts to capture subtle visual differences using sparse/low-rank analysis:

Unsupervised Feature Extraction Inspired by Latent Low-Rank Representation Latent Low-Rank Representation (Lat LRR) has the empirical capability of identifying "salient" features. However, the reason behind this feature extraction effect is still not understood. Its optimization leads to non-unique solutions and has high computational complexity, limiting its potential in practice. We show that Lat LRR learns a transformation matrix which suppresses the most significant principal components corresponding to the largest singular values while preserving the details captured by the components with relatively smaller singular values. Based on this, we propose a novel feature extraction method which directly designs the transformation matrix

and has similar behavior to Lat LRR. Our method has a simple analytical solution and can achieve better performance with little computational cost. The effectiveness and efficiency of our method are validated on two face recognition datasets.

Face recognition can be regarded as ultra fine-grained recognition tasks with less viewpoint variation. In terms of regular fine-grained recognition with larger viewpoint variation and deformation, an intuitive way is to consider similar sparse/low-rank techniques in patch-level, since smaller patches are more robust to viewpoint/deformation. However, limited progress has been made in this direction. The main reason might be that sparse/low-rank analysis assumes that a visual descriptor can be represented by linear combination of a code book of descriptors, and such linear assumption might not hold for objects more complex than rectified faces. This lesson inspired us to avoid elegant theoretical background and develop patch-based framework in a more intuitive way which leads to the work of Chapter 3.

In **Chapter 3**, we address the problem of fine-grained recognition via a patch-based framework built upon Convolutional Neural Network (CNN) features:

Mining Discriminative Triplets of Patches for Fine-Grained Classification Fine-grained classification involves distinguishing between similar sub-categories based on subtle differences in highly localized regions; therefore, accurate localization of discriminative regions remains a major challenge. We describe a patch-based framework to address this problem. We introduce triplets of patches with geometric constraints to improve the accuracy of patch localization, and automatically mine discriminative geometrically-constrained triplets for classification. The resulting ap-

proach only requires object bounding boxes. Its effectiveness is demonstrated using four publicly available fine-grained datasets, on which it outperforms or obtains comparable results to the state-of-the-art in classification.

Our multi-stage framework using CNN features achieves better performance compared to two types of baselines: *(i)* it outperform its counterpart with hand-crafted features (*e.g.*, HOG) by a huge margin, which means that low-level CNN features are far more effective than previous hand-crafted ones; *(ii)* it significantly outperforms the baseline which finetune the same CNN used for feature extraction. This further suggests that CNN's ability to learn mid-level representations is limited and still has sufficient room to improve. Based on these observations, we have done extensive works in Chapter 4 and Chapter 5 to *enhance the mid-level representation learning capability of CNN*.

Chapter 4 proposes an intuitive way to introduce supervision to intermediate layers to improve their discriminativeness:

Learning Discriminative Features via Label Consistent Neural Network Deep Convolutional Neural Networks (CNN) enforces supervised information only at the output layer, and hidden layers are trained by back propagating the prediction error from the output layer without explicit supervision. We propose a supervised feature learning approach, Label Consistent Neural Network, which enforces direct supervision in late hidden layers. We associate each neuron in a hidden layer with a particular class label and encourage it to be activated for input signals from the same class. More specifically, we introduce a label consistency regularization called

"discriminative representation error" loss for late hidden layers and combine it with classification error loss to build our overall objective function. This label consistency constraint alleviates the common problem of gradient vanishing and tends to faster convergence; it also makes the features derived from late hidden layers discriminative enough for classification even using a simple k-NN classifier, since input signals from the same class will have very similar representations. Experimental results demonstrate that our approach achieves state-of-the-art performances on several public benchmarks for action and object category recognition.

In **Chapter 5**, we proposes a more sophisticated and effective end-to-end network specifically designed for fine-grained recognition, which learns discriminative patches within a CNN. Compared with the work in Chapter 4, it is both more effective and more human-interpretable:

Learning a Discriminative Filter Bank within a CNN for Fine-grained Recognition Compared to earlier multistage frameworks using CNN features, recent end-to-end deep approaches for fine-grained recognition essentially enhance the mid-level learning capability of CNNs. Previous approaches achieve this by introducing an auxiliary network to infuse localization information into the main classification network, or a sophisticated feature encoding method to capture higher order feature statistics. We show that mid-level representation learning can be enhanced within the CNN framework, by learning a bank of convolutional filters that capture class-specific discriminative patches without extra part or bounding box annotations. Such a filter bank is well structured, properly initialized and discriminatively learned through

a novel asymmetric multi-stream architecture with convolutional filter supervision and a non-random layer initialization. Experimental results show that our approach achieves state-of-the-art on three publicly available fine-grained recognition datasets (CUB-200-2011, Stanford Cars and FGVC-Aircraft). Ablation studies and visualizations are provided to understand our approach.

After obtaining satisfactory results with CNN framework incorporating patch-level discriminative information, we explore the possible benefits of incorporating 3D information into fine-grained recognition. Limited success has been achieved using 3D information to help 2D recognition and the observation is that, with current strong deep neural network architecture and large amount of training data, 2D appearance information seems sufficient for the recognition task. During exploration, we found a related interesting problem of estimating the 3D pose for fine-grained object categories. This category-based pose estimation problem can be applied to practical scenario of vehicle damage assessment, which registers the fine-grained 3D model with the 2D car image as the basis for further assessment.

Therefore, following recent trends in visual recognition, we goes beyond category label of fine-grained objects in **Chapter 6**:

Continuous 3D Pose Estimation for Fine-Grained Objects Continuous 3D object pose estimation from monocular images recently achieves attention in the computer vision community. With enough number of training data, the deep Convolutional Neural Networks (CNNs) are able to learn discriminative features to identify the 3D pose of an object, even with a single image as input. However, due to the expensive

cost of obtaining high-quality continuous 3D annotations for objects in real images, most 3D pose estimation dataset are limited to a small amount. Moreover, all existing datasets are related to generic object types and there is so far no dataset for fine-grained objects. In this work, we introduce a new large dataset that is able to benchmark both fine-grained object recognition and 3D object pose estimation. Specifically, we augment the three existing popular fine-grained recognition dataset (Stanford Cars, FGVC-Aircraft and CompCars), and annotate each instance in the image with the corresponding ground truth 3D pose. We then study the multi-task problem of simultaneous fine-grained object recognition and 3D pose estimation. To achieve this, we design a new network architecture by modifying the recent state-of-the-art Mask R-CNN and apply it to the multi-task problem. To utilize the 3D information, we introduce a novel 3D representation, named as location field, that we find is very efficient for representing 3D shapes. With the new 3D dataset and representation, We compare our model with different network structures and obtain state-of-the-art results on continuous 3D object pose estimation for fine-grained objects. The new datasets will be released upon acceptance.

1.2.1 Publications

The work in Chapter 2 was accepted by *WACV 2015*; the work in Chapter 3 was accepted by *CVPR 2016*; the work in Chapter 4 was accepted by *WACV 2017*. The work in Chapter 5 was accepted by *CVPR 2018*. The work in Chapter 6 is currently in preparation for a future submission.

Chapter 2: Unsupervised Feature Extraction Inspired by Latent Low-Rank Representation

2.1 Background and Motivation

Recently, Low-Rank Representation (LRR) [12–16] has attracted attention in the field of unsupervised subspace segmentation [17], since it can effectively cluster high-dimensional data into low-dimensional subspaces by learning the lowest rank representation of the data matrix. Among various versions of LRR, Latent Low-Rank Representation (LatLRR) [1] has the novel ability to extract "salient features" from visual data, which was interpreted as the deviation of each sample from the "principal features".

LatLRR solves the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* + \|L\|_* + \lambda\|E\|_1, \\ \text{s.t.} \quad & X = XZ + LX + E, \end{aligned} \tag{2.1}$$

where X is the data matrix with each column being one data sample and $\|\cdot\|_*$ denotes the nuclear norm. In [1], the authors empirically found out that using the second term, LX, for classification can significantly improve accuracy, thanks

to its "salient features". Although LatLRR originally aimed to improve subspace clustering and the salient feature extraction effect was just a by-product, it outperformed many mainstream dimensionality reduction techniques, such as Principal Component Analysis (PCA) [18], Neighborhood Preserving Embedding (NPE) [19], Locality Preserving Projection (LPP) [20] and Nonnegative Matrix Factorization (NMF) [21].

While LatLRR has shown promissing results, it suffers from the following problems.

- An explanation for its observed capability to identify "salient" features is unknown, which constrains its potential.
- It has been observed that the solution to (2.1) is not unique, which potentially reduces its reliability. Zhang *et al.* [22, 23] have derived a closed form solution to the noiseless version of (2.1), but the reliability problem for "salient" feature extraction remains.
- The complexity of LatLRR depends on the dimensionality of the feature vectors [1]. Previous improvements to LatLRR have actually increased the complexity of the optimization, such as the introduction of a more complex objective function [24–26] or more complex constraints [27], which further increases the computational burden.

We provide an explanation of LatLRR's feature extraction effect. Based on this, we propose a new and computationally simpler feature extraction method. The contributions of this work are twofold.

- We show that the singular values of L learnt by LatLRR have a reweighting effect, which suppresses the most significant principal components of data matrix X, while highlighting the detailed information carried by the components corresponding to X's relatively smaller singular values.

- Using our characterization of the solutions produced by LatLRR, we design a new feature extraction method that produces solutions similar to LatLRR, but with a simple unique analytical solution. Our method outperforms LatLRR by computing a single SVD decomposition, and thus can be applied to higher dimensional data.

2.2 Analysis of LatLRR's Feature Extraction

In this section, we interpret the feature extraction effect of LatLRR, mainly from the perspective of SVD decompostion.

Suppose X and L have a skinny SVD

$$X = U_X \Sigma_X V_X^{\mathrm{T}} \quad \text{and} \quad L = U_L W_L V_L^{\mathrm{T}}. \tag{2.2}$$

The meaning of "skinny" is that Σ_X and W_L are square matrices of size $\mathrm{rank}(X)$ and $\mathrm{rank}(L)$ respectively, only containing non-zero singular values.

Let l_i be the i^{th} singular value of L, and v_{Li} (u_{Li}) the i^{th} column of V_L (U_L).

Consider the effect of L operating on a single data sample x_0:

$$Lx_0 = U_L W_L V_L^{\mathrm{T}} x_0 = \sum_{i=1}^{r} l_i \langle v_{Li}, x_0 \rangle u_{Li}, \tag{2.3}$$

where $r = \mathrm{rank}(L)$. According to the definition of SVD, by multiplying by L, the data has been projected onto each column of V_L and weighted by the singular values. Then the weighted projections are used as the coefficients of $\{u_{Li}\}$, a subset of an orthonormal basis of L's column space. This interpretation of SVD decomposition lays the foundation of our analysis below.

2.2.1 Preliminary

Recently, Zhang *et al.* [22] found that the solution to LatLRR is not unique. Moreover, they derived a closed form solution for the noise free LatLRR

$$\text{minimize} \quad \|Z\|_* + \|L\|_*, \quad \text{s.t.} \quad X = XZ + LX. \tag{2.4}$$

We restate the main result of [22] on noise free LatLRR in the following theorem.

Theorem 1. *The complete solutions to problem (2.4) must be of the following form*

$$Z = V_X W_Z V_X^{\mathrm{T}} \quad \textit{and} \quad L = U_X (I - W_Z) U_X^{\mathrm{T}}, \tag{2.5}$$

where W_Z is any block diagonal matrix satisfying: 1. if $[\Sigma_X]_{ii} \neq [\Sigma_X]_{jj}$ then $[W_Z]_{ij} =$

0*; 2. both* W_Z *and* $I - W_Z$ *are positive semi-definite [22].*

Notice that in practice the singular values of X are usually distinct, therefore W_Z becomes a diagonal matrix $\text{diag}\{z_1, z_2, ..., z_r\}$ with $0 \leq z_i \leq 1$ for all i. Then (2.5) becomes equivalent to (2.2), *i.e.*

$$U_L = V_L = U_X, \quad W_L = I - W_Z, \tag{2.6}$$
$$r = \text{rank}(L) = \text{rank}(X),$$

and the effects of Z and L on X become

$$\begin{aligned} XZ &= U_X \Sigma_X W_Z V_X^{\mathrm{T}} \\ &= U_X \begin{bmatrix} z_1\sigma_{X1} & & \\ & \ddots & \\ & & z_r\sigma_{Xr} \end{bmatrix} V_X^{\mathrm{T}} \end{aligned} \tag{2.7}$$

$$\begin{aligned} LX &= U_X (I - W_Z) \Sigma_X V_X^{\mathrm{T}} \\ &= U_X \begin{bmatrix} l_1\sigma_{X1} & & \\ & \ddots & \\ & & l_r\sigma_{Xr} \end{bmatrix} V_X^{\mathrm{T}}, \end{aligned} \tag{2.8}$$

where $l_i = 1 - z_i$, σ_{Xi} is the i^{th} largest singular value of X.

Moreover, (2.3), the effect of L on a single data x_0 becomes

$$Lx = \sum_{i=1}^{r} l_i \langle u_{Xi}, x_0 \rangle u_{Xi}, \tag{2.9}$$

i.e. the data is projected onto the components $\{u_{Xi}\}$ and weighted by $\{l_i\}$.

From (2.8) and (2.9), as will be shown in Section 2.2.2 and Section 2.2.3, $W_L = I - W_Z$ suppresses the components corresponding to very large σ_{Xi}, and preserves the ones corresponding to relatively small σ_{Xi}. Since the orthogonal matrices in (2.5) are determined as U_X and V_X, the numerical optimization with respect to Z and L is essentially learning the singular values $\{z_i\}$ and $\{l_i\}$, resulting in a solution from the solution set defined by (2.5). This is exactly what LatLRR is accomplishing.

In the rest of this section, we gain insight from an empirical result in Section 2.2.2, and then theoretically explore the source of the empirical results in Section 2.2.3.

2.2.2 Empirical Analysis

The following phenomenon can provide some insight into LatLRR's feature extraction capability. Our empirical results suggest that different optimization ordering leads to different reweightings of principal components, which strongly affects the performance.

In the iterative algorithm described in [1] (the noise free version summarized in Algorithm 1), within each iteration, Z is updated before L. However, if we update L before Z in each iteration (exchange Step 3 with Step 4 in Algorithm 1), the

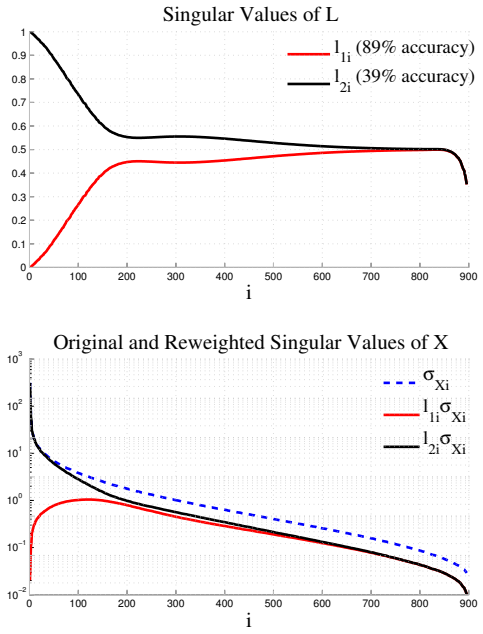

Figure 2.1: The optimization ordering that yields the best performance (red) down-weights the most significant principal components (low i), while the other ordering (black) gives them more weight. Top: the singular values of transformation matrix L. Bottom: the singular values of data matrix X and transformed data LX. The latter is displayed in a semi-log graph due to the large variations in scale.

final classification performance decreases significantly (*e.g.*from approximately 89% to 39% on Extended Yale Database B).

Let $\{l_{1i}\}$ be the singular values of L learnt by the original Algorithm 1, and $\{l_{2i}\}$ the singular values learnt with the exchanged steps. The singular values of L, X, and LX are displayed in Figure 2.1 for both cases. For the successful optimization ordering, the u_{Li} with the largest σ_{Xi} and largest $l_{1i}\sigma_{Xi}$ are displayed in Figure 2.2.

As can be seen from Figure 2.1 and (2.9), $\{l_{1i}\}$ plays the role of reweighting, which suppresses the components corresponding to the first several largest σ_{Xi}'s, and thus highlights those containing detailed information. On the other hand, however, l_{2i} put relative small weights on the basis corresponding to small σ_{Xi} and mainly preserves the information carried by components with very large σ_{Xi}, which significantly reduces the performance. Therefore, the ability of LatLRR to extract features is related to the weighting effect of $\{l_i\}$.

Our analysis is consistent with the observation made by [28], that dropping the first three principal components in PCA can effectively improve the classification accuracy of faces. [28] claimed that the first a few principal components might mainly capture the variations caused by photometric factors such as illumination and shadow, therefore removing those irrelevant variations might account for the effectiveness of such practical technique. LatLRR, which reweights the components, can be regarded as a "soft version" of dropping the first few components.

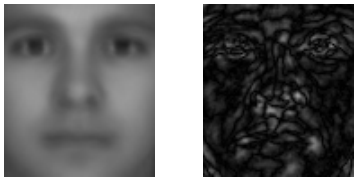

Figure 2.2: An example principal component (left) and the component corresponding to relatively small σ_{Xi} (right). On the left is the u_{Xi} corresponding to the largest σ_{Xi}, and on the right is the one corresponding to the largest $l_{1i}\sigma_{Xi}$ after reweighting.

Algorithm 1 Solving Problem (2.4) by Inexact ALM

Initialize: $Z_0 = J_0 = 0$, $L_0 = S_0 = 0$, Y_{10} $= Y_{20} = Y_{30} = 0$, $\mu_0 > 0$. Set parameters (ρ and ϵ).
while not converged **do**
1. Fix others and update J by setting $J_{k+1} =$ $\operatorname{argmin}_J \frac{1}{\mu_k}\|J\|_* + \frac{1}{2}\|J - (Z_k + Y_{2k}/\mu_k)\|_{\mathrm{F}}^2$.
2. Fix others and update S by setting $S_{k+1} =$ $\operatorname{argmin}_S \frac{1}{\mu_k}\|S\|_* + \frac{1}{2}\|S - (L_k + Y_{3k}/\mu_k)\|_{\mathrm{F}}^2$.
3. Fix others and update Z by setting $Z_{k+1} = (I + X^{\mathrm{T}}X)^{-1}(X^{\mathrm{T}}(X - L_kX) + J_{k+1}$ $+(X^{\mathrm{T}}Y_{1k} - Y_{2k})/\mu_k)$.
4. Fix others and update L by setting $L_{k+1} = ((X - XZ_{k+1})X^{\mathrm{T}} + S_{k+1} + (Y_{1k}X^{\mathrm{T}}$ $-Y_{3k})/\mu_k)(I + XX^{\mathrm{T}})^{-1}$
5. Update the mulipliers by $Y_{1(k+1)} = Y_{1k} + \mu_k(X - XZ_{k+1} - L_{k+1}X)$, $Y_{2(k+1)} = Y_{2k} + \mu_k(Z_{k+1} - J_{k+1})$, $Y_{3(k+1)} = Y_{3k} + \mu_k(L_{k+1} - S_{k+1})$.
6. Update μ by $\mu_{k+1} = \rho\mu_k$.
7. Check the convergence conditions: $\|X - XZ_{k+1} - L_{k+1}X\|_\infty < \epsilon$, $\|Z_{k+1} - J_{k+1}\|_\infty < \epsilon$, and $\|L_{k+1} - S_{k+1}\|_\infty < \epsilon$.
end while

2.2.3 Theoretical Analysis

In this subsection, we analyze the inexact Augmented Lagrange Multiplier (ALM) method [29] adopted by [1], for the noise free LatLRR (2.4), and derive a closed-form approximate solution that has the same effect as ALM on component weighting. Our analysis explains our empirical observations above, where the singluar values of L suppress the most pincipal u_{Xi}'s corresponding to the largest σ_{Xi}'s, while preserving the components corresponding to relatively smaller σ_{Xi}'s. In the rest of this work, we will keep using the letter i to denote the index of singular values, while we use letter k to indicate the number of iterations.

LatLRR's Algorithm Overview

The inexact ALM method is outlined in Algorithm 1. Step 1 and 2 are solved by singular value thresholding operator [30], *i.e.*

$$\begin{aligned}\mathcal{D}_{1/\mu}(Y) &= \arg\min_X \left\{ \frac{1}{\mu}\|X\|_* + \frac{1}{2}\|X-Y\| \right\}, \\ &= U_Y \mathrm{diag}\left\{ \max\left(0, \sigma_{Yi} - \frac{1}{\mu}\right) \right\} V_Y^{\mathrm{T}}, \end{aligned} \tag{2.10}$$

where σ_{Yi} is the i^{th} singular value of Y, whose SVD is

$$Y = U_Y \mathrm{diag}\left\{\sigma_{Yi}\right\} V_Y^{\mathrm{T}}.$$

The only assumption made to simplify the analysis is that ρ is relatively large. The form of the solution is first given by Proposition 1 and then the specific solution

is derived from Proposition 2 and 3. Proofs of Propositions 1-3 are provided in the supplementary material.

Simplification of the Analysis

Proposition 1. *During the iteration procedure described in Algorithm 1, Z_k and L_k always keep the form*

$$Z_k = V_X W_{Zk} V_X^{\mathrm{T}}, \quad L_k = U_X W_{Lk} U_X^{\mathrm{T}}, \tag{2.11}$$

$\forall$ *positive integer k. Where W_{Zk} and W_{Lk} are $r \times r$ diagonal matrices containing the singular values.*

Furthermore, the orthogonal matrices involved in the SVD decomposition of any matrix in Algorithm 1 must be U_X or V_X, depending on its shape.

Proposition 1 can be proved easily by induction. It suggests that we can simplify the analysis by only focusing on the singular values of the matrices. Furthermore, in Algorithm 1, singular values with the same index are modified together, independent from those with different indices, therefore we omit the index i and only focus on the iteration number k. Let the lower case letters denote the singular values of their corresponding matrices in upper case letters. For instance, $z_k = \{z_{ki}, i = 1, 2, ..., r\}$ and $l_k = \{l_{ki}, i = 1, 2, ..., r\}$ denote the singular values of Z_k and L_k, respectively, and $z_k + l_k = \{z_{ki} + l_{ki}, i = 1, 2, ..., r\}$. Hence, the iteration procedure in Algorithm 1 is equivalent to Table 2.1.

Under the assumption that ρ is relatively large, we can simplify the analysis by omitting the last term on the right side of Step 3 and 4 in Table 2.1, since it can

Algorithm 1 (Simplified)

Initialize: $z_0 = j_0 = l_0 = s_0 = y_{10} = y_{20} = y_{30} = 0, \mu_0 > 0.$
while not converged **do**
1. $j_{k+1} = \max\{0, z_k + y_{2k}/\mu_k - 1/\mu_k\}$
2. $s_{k+1} = \max\{0, l_k + y_{3k}/\mu_k - 1/\mu_k\}$
3. $z_{k+1} = \frac{1}{1+\sigma_X^2}(\sigma_X^2(1 - l_k) + j_{k+1}) + \frac{\sigma_X y_{1k} - y_{2k}}{(1+\sigma_X^2)\mu_k}$
4. $l_{k+1} = \frac{1}{1+\sigma_X^2}(\sigma_X^2(1 - z_{k+1}) + s_{k+1}) + \frac{\sigma_X y_{1k} - y_{3k}}{(1+\sigma_X^2)\mu_k}$
5. $y_{1(k+1)} = y_{1k} + \mu_k \sigma_X (1 - z_{k+1} - l_{k+1})$
$y_{2(k+1)} = y_{2k} + \mu_k (z_{k+1} - j_{k+1})$
$y_{3(k+1)} = y_{3k} + \mu_k (l_{k+1} - s_{k+1})$
6. $\mu_{k+1} = \rho \mu_k$
7. Check the convergence conditions.
end while

Table 2.1: The simplification of Algorithm 1 by focusing on the operations of the singular values.

be proved by induction that

$$y_{1k} = \sigma_X \sum_{t=1}^{k} \mu_{t-1} e_{1t}, \quad y_{2k} = \sum_{t=1}^{k} \mu_{t-1} e_{2t},$$
$$y_{3k} = \sum_{t=1}^{k} \mu_{t-1} e_{3t}, \quad \mu_k = \rho^k \mu_0, \tag{2.12}$$

where $e_{1t} = 1 - z_t - l_t,\ e_{2t} = z_t - j_t,\ e_{3t} = l_t - s_t.$

Then the last term on the right side of Step 3 becomes

$$\begin{aligned}
&\frac{\sigma_X y_{1k} - y_{2k}}{(1+\sigma_X^2)\mu_k} \\
=&\sum_{t=1}^{k} \frac{\mu_{t-1}}{\mu_k} \left(\frac{\sigma_X^2}{1+\sigma_X^2} e_{1t} - \frac{1}{1+\sigma_X^2} e_{2t} \right) \\
=&\sum_{t=1}^{k} \left(\frac{1}{\rho}\right)^{k+1-t} \left(\frac{\sigma_X^2}{1+\sigma_X^2} e_{1t} - \frac{1}{1+\sigma_X^2} e_{2t} \right) \\
<&e_{\max} \sum_{t=1}^{+\infty} \left(\frac{1}{\rho}\right)^{t} = \frac{e_{\max}}{\rho - 1}
\end{aligned} \tag{2.13}$$

In practice $e_{\max}$ is very small. Therefore (2.13) approaches zero when ρ is relatively large, and omitting it can provide a simple and good approximation. Through similar analysis by replacing y_{2k} with y_{3k}, we can obtain the same conclusion for the last term of Step 4.

Since Step 1 and 2 in Table 2.1 perform thresholding according to the value of $1/\mu_k$, we divide the analysis into 2 stages: large $1/\mu_k$ when μ_k is very small at the beginning, and small $1/\mu_k$ when μ_k becomes very large by the end of the iteration.

The Closed-Form Solution

When μ_k is very small, $1/\mu_k$ is so large that $j_{k+1} = s_{k+1} = 0$. Combine the approximation upon Step 3 and 4 discussed above, and thus the iteration procedure is equivalent to

$$z_{k+1} = \alpha(1 - l_k), \quad l_{k+1} = \alpha(1 - z_{k+1}), \tag{2.14}$$

where

$$\alpha = \frac{\sigma_X^2}{1+\sigma_X^2} = 1 - \frac{1}{1+\sigma_X^2}, \tag{2.15}$$

Solving the linear recursive sequence (2.14) gives the following proposition.

Proposition 2. *Assuming ρ is relatively large, when μ_k is small, the iteration procedure in Table 2.1 is approximately equivalent to (2.14), and the solution after the k^{th} iteration is as follows.*

$$z_k = \frac{\alpha + \alpha^{2k}}{1+\alpha}, \quad l_k = \alpha(1-z_k) = \frac{\alpha^3 + \alpha^{2k+1}}{1+\alpha}, \tag{2.16}$$

where α is defined by (2.15).

When μ_k is very large, however, $1/\mu_k \approx 0$ so that

$$\begin{aligned} j_{k+1} &= \max\{0, z_k + y_{2k}/\mu_k - 1/\mu_k\} \\ &= z_k + y_{2k}/\mu_k - 1/\mu_k \approx z_k \end{aligned} \tag{2.17}$$

Similarly, it follows that

$$s_{k+1} \approx l_k. \tag{2.18}$$

Plugging in (2.17) and (2.18) into Step 3 and 4, and adopting the same approximation as that of Proposition 2, we obtain an equivalent procedure for the

large-μ_k case.

$$z_{k+1} = \alpha(1 - l_k) + (1 - \alpha)z_k, \tag{2.19}$$
$$l_{k+1} = \alpha(1 - z_{k+1}) + (1 - \alpha)l_k,$$

Proposition 3. *Assuming ρ is relatively large, when μ_k is large, the iteration procedure in Table 2.1 is approximately equivalent to (2.19). When the iteration terminates, the final results z and l satisfy*

$$z \to \frac{(1 - \alpha)z_{k_0} - l_{k_0} + 1}{2 - \alpha}, \quad l \to 1 - z. \tag{2.20}$$

where α is defined by (2.15), and k_0 is some starting point from which the large-μ_k condition holds.

When ρ is relatively large, the transient state between the two stages only lasts a very short time. Therefore, omitting it can provide a good and simple approximation to the solution. We pick up a dividing point k_0 of the two stages, use (2.14) to approximate the iterations when $k \leq k_0$, and use (2.19) to approximate those when $k > k_0$. Plugging (2.16) into z_{k_0} and l_{k_0} of (2.20), we conclude that, when the iteration terminates,

$$z \to 1 - \frac{1 - \alpha^{2k_0}}{(2 - \alpha)(1 + \alpha)}, \quad l \to \frac{1 - \alpha^{2k_0}}{(2 - \alpha)(1 + \alpha)}, \tag{2.21}$$

where α is defined by (2.15) and k_0 is the dividing point of the two stages.

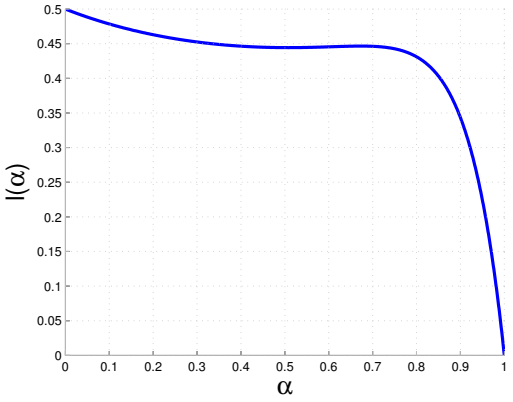

Figure 2.3: l as a function of α given by (2.21). The definition of α is given in (2.15). For small σ_X, α approaches 0 and l decreases very slowly from 1/2; for very large σ_X, α approaches 1 and l drops very quickly. Parameter settings: $k_0 = 6$.

According to (2.15), α is monotonically increasing as σ_X increases from 0 to infinity. Therefore, when σ_X approaches 0, α also approaches 0 and l will approach 1/2; when σ_X approaches infinity, α approaches 1 and l will approach 0. l as a function of α is displayed in Figure 2.3. As can be seen, for relatively small α, which corresponds to small σ_X, l decreases very slowly from 1/2; for α close to 1, which corresponds to very large σ_X, l drops very quickly to a small value.

Result Evaluation

According to (2.21), our theoretical result for the i^{th} singular value of L becomes

$$l_i = \frac{1 - \alpha_i^{2k_0}}{(2 - \alpha_i)(1 + \alpha_i)}, \tag{2.22}$$

where

$$\alpha_i = \frac{\sigma_{Xi}^2}{1 + \sigma_{Xi}^2} = 1 - \frac{1}{1 + \sigma_{Xi}^2}. \tag{2.23}$$

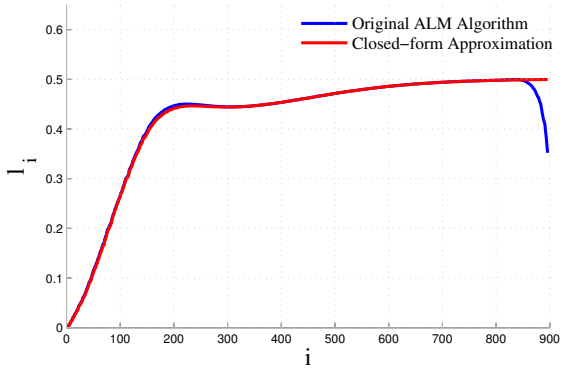

Figure 2.4: The singular values of our theoretical approximation (red) compared to those achieved by ALM in practice (blue), where l_i denotes the i^{th} singular value of L. Our theoretical approximation very closely models the behavior of ALM. Parameter settings: $\mu_0 = 10^{-6}, \rho = 5$, $\epsilon = 10^{-4}$ (practice); $k_0 = 6$ (theory).

The comparison between the reweighting behavior of the L matrix according to theoretical approximation and actual result of ALM is displayed in Figure 2.4. We plot the singular values of L obtained in practice by Algorithm 1 (displayed in blue) as well as that calculated theoretically by (2.22) (displayed in red). As can be seen, the theoretical approximation convincingly explains the behavior of Algorithm 1, and the minor error occuring at the tail is mainly due to the fact that the stopping criterion of the theoretical analysis is the limit condition of ALM.

From (2.8) and (2.21), we conclude that L can suppress principal components corresponding to very large σ_{Xi}'s, by putting near-zero weights on them through its singular values l_i's. On the other hand, for those components corresponding to relatively small σ_{Xi}'s, multiplying by L will not affect them too much since the corresponding l_i's are almost constant.

2.3 Unsupervised Feature Extraction Inspired by Latent Low-rank Representation

We can now directly design a transformation matrix W_Z that behaves similarly to the original LatLRR algorithm. After W_Z has been obtained, with Z and L constructed according to (2.5), X can be decomposed into a principal part XZ and a detailed part LX. Then following the approach of LatLRR [1], LX can be used for classification.

Concretely, the problem boils down to designing the objective function f for the following problem

$$\begin{aligned} \text{minimize} \quad & f \\ \text{s.t.} \quad & X = XZ + LX \\ & Z = V_X W_Z V_X^{\mathrm{T}},\ L = U_X (I - W_Z) U_X^{\mathrm{T}}, \\ & W_Z = \mathrm{diag}(z_1, z_2, ..., z_r), 0 \leq z_i \leq 1\ \forall i, \end{aligned} \tag{2.24}$$

such that $(I - W_Z)$ down-weights the most significant principal components while preserving the others. To achieve this goal, a suitable objective function should at least satisfy the following two properties

(i) XZ must contain most information of X, *i.e.* the error $X - XZ$ cannot be too large.

(ii) XZ is only allowed to contain the most principal features, which means that

the columns of XZ must be similar to each other. Thus $\|Z\|_*$ cannot be large, since the nuclear norm reflects low-rank self-expressiveness, which is a very good similarity measure for multiple samples [17].

Attempting to balance (i) and (ii), a natural objective function is

$$f = \|Z\|_* + \lambda \|X - XZ\|_{\mathrm{F}}^2, \tag{2.25}$$

where λ is a trade-off parameter, which is expected to be small, considering the scale of the error.

Plugging (2.25) into (2.24), the formulation becomes

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* + \lambda \|X - XZ\|_{\mathrm{F}}^2, \\ \text{s.t.} \quad & X = XZ + LX \\ & Z = V_X W_Z V_X^{\mathrm{T}}, \ L = U_X (I - W_Z) U_X^{\mathrm{T}}, \\ & W_Z = \mathrm{diag}(z_1, z_2, ..., z_r), 0 \leq z_i \leq 1 \ \forall i. \end{aligned} \tag{2.26}$$

Since the z_i's are the only independent variables of problem (2.26), we can eliminate other variables by using singular values to express the norms, which results in the following equivalent problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{r} z_i + \lambda \sum_{i=1}^{r} (1 - z_i)^2 \sigma_{Xi}^2 \\ \text{s.t.} \quad & 0 \leq z_i \leq 1 \quad \forall i. \end{aligned} \tag{2.27}$$

Problem (2.27) can be solved analytically as

$$w_i^* = \begin{cases} 1 - \frac{1}{2\lambda\sigma_{Xi}^2}, & \sigma_{Xi} \geq \sqrt{\frac{1}{2\lambda}} \\ 0, & \sigma_{Xi} < \sqrt{\frac{1}{2\lambda}} \end{cases} \tag{2.28}$$

The result can be interpreted as follows. For a small σ_{Xi}, since the information it adds to XZ is too detailed, its negative contribution to the error is smaller than its positive contribution to the nuclear norm, therefore it is filtered out by $z_i = 0$. On the other hand, the larger σ_{Xi} is, the larger z_i will be. This means that the most significant principal components have been preserved.

In contrast,

$$l_i^* = 1 - w_i^* = \min\left\{\frac{1}{2\lambda\sigma_{Xi}^2},\ 1\right\} \tag{2.29}$$

extracts features corresponding to small σ_{Xi}'s. Our feature extraction procedure is summarized in Table 2.2.

Given training data X_{train}
$[U_X, \text{diag}\{\sigma_{Xi}\}, V_X] = \text{svd}(X_{\text{train}})$ $l_i^* = \min\left\{\frac{1}{2\lambda\sigma_{Xi}^2},\ 1\right\}$ $L^* = U_X \text{diag}\{l_i^*\} U_X^{\text{T}}$
Use $L^* X_{\text{train}}$ and $L^* X_{\text{test}}$ for classification.

Table 2.2: Our feature extraction procedure.

2.4 Experiments

In this section we evaluate both the performance and efficiency of our method. We mainly compared our approach with LatLRR, since it has already been reported in [1] that LatLRR outperforms dimensionality deduction methods such as Locality Preserving Projection (LPP) [20], Neighborhood Preserving Embedding (NPE) [19] and Nonnegative Matrix Factorization (NMF) [21] with a large margin (see Table 2 of [1]).

Datasets We tested our feature extraction using both the Extended Yale Database B [31] and CMU PIE face databases [32], two common datasets for face recognition. Extended Yale B consists of 2414 frontal face images of 38 individuals, and each individual has approximately 64 images. For CMU PIE face databases, the subset of frontal faces (referred to as C27) with different illumination and facial expressions was used, which contains 3329 images of 68 individuals.

Experimental Settings For fair comparison, we adopted the same settings as [1] when conducting performance test on Extended Yale Database B. Each image was resized to 32×28 and reshaped into a data vector of dimension 896, whose entries were normalized to $[0, 1]$. 47% of the randomly split data was used for training and the rest for testing. After Z^* and L^* were learnt, only L^*X_{train} and L^*X_{test} were fed into the K-nearest neighbor classifier (K-NN) based on Euclidean distance. We implemented and measured our own method with the trade-off parameter $\lambda = 0.02$, while copying the results of LatLRR from [1].

When conducting the performance test on PIE, the settings remained the same except for the following: we resized each image to 32×32 and used 33% of the data for training; we ran the implementation from the author of [1] to test LatLRR.

To test the efficiency of our method, we recorded the running time of our method followed by 1-NN classification and compared it with that of LatLRR. This experiment was performed on Extended Yale Database B, using a machine with two Intel(R) Xeon(R) E5603 @ 1.6GHz.

Results and Analysis The results of performance tests are displayed in Table 2.3 and Table 2.4. As can be seen, our method has similar behavior to LatLRR: it largely outperforms the baseline of "Raw Data", and the product L^*X is a suitable input for dimensionality reduction techniques such as PCA. In Figure 2.5, we plot the singular values $\{l_i^*\}$ of L^* calculated by (2.29), and compare with those learnt by LatLRR. From Figure 2.5, it becomes clear that our method reaps the benefit of weighting effect discussed in Section 2.2. Moreover, since our method is specifically designed for feature extraction, it further outperforms LatLRR, which was originally designed for subspace clustering. Some examples of using our method to extract detailed features are displayed in Figure 2.6.

The results of the efficiency test are displayed in Table 2.5. Since our method only requires a single SVD decomposition, it has an overwhelming advantage over LatLRR when the dimensions of the feature vectors are the same. With such efficiency, our method can be applied to higher dimensional data.

Brief Summary Specifically designed for feature extraction, our method can

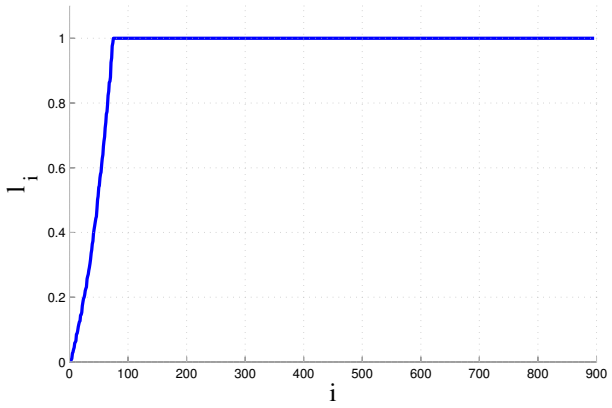

Figure 2.5: The singular values of L learnt by our method, where l_i denotes the i^{th} singular value. By comparing with Figure 2.4, we can see that our method has the reweighting effect similar to LatLRR.

	Raw Data	Raw Data+PCA (317D)	LatLRR	LatLRR+PCA (400D)	Ours	Ours+PCA (400D)
1-NN	61.07	61.54	88.76	87.28	93.72	93.09
3-NN	59.81	60.03	87.76	85.95	94.27	93.49
5-NN	58.16	58.54	86.03	85.87	93.88	93.56

Table 2.3: Classification accuracies (%, averaged over 20 runs) on Extended Yale Database B. For fair comparison, the results related to raw data and LatLRR are cited from [1], who chose the dimension 317D to obtain the best result within the range of 400D.

	Raw Data	Raw Data+PCA (317D)	LatLRR	LatLRR+PCA (400D)	Ours	Ours+PCA (400D)
1-NN	77.52	77.48	94.83	94.83	96.25	96.25
3-NN	71.32	71.32	93.71	93.71	96.12	96.16
5-NN	63.25	63.16	91.21	90.82	96.08	96.03

Table 2.4: Classification accuracies (%, averaged over 20 runs) on CMU PIE face databases.

achieve better performance than LatLRR with little computational cost, and can be applied to higher dimensional data effectively.

Image Size	Running Time(s) LatLRR	Accuracy(%) LatLRR	Running Time(s) Ours	Accuracy(%) Ours
32×28(896D)	160.26	88.78	4.21	93.72
48×42(2016D)	458.28	88.50	10.69	93.38
96×84(8064D)	1097.48	85.61	44.93	94.04

Table 2.5: Results of efficiency test (averaged over 20 runs) on Extended Yale Database B.

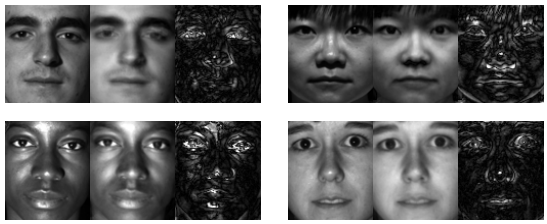

Figure 2.6: The visualization of the decomposition by our method. In each group of the same individual, the original data X (left) is decomposed into a principal part XZ^* (middle) and a detailed part L^*X (right).

Chapter 3: Mining Discriminative Triplets of Patches for Fine-Grained Classification

3.1 Motivation

The task of fine-grained classification is to recognize sub-ordinate categories belonging to the same super-ordinate category [33–36]. The major challenge is that fine-grained objects share similar overall appearance and only have subtle differences in highly localized regions. To effectively and accurately find these discriminative regions, some previous approaches utilize humans-in-the-loop [37–39], or require semantic part annotations [40–46] or 3D models [36,47]. These methods are effective, but they require extra keypoint/part/3D annotations from humans, which are often expensive to obtain. On the other hand, recent research on discriminative mid-level visual elements mining [48–52] automatically finds discriminative patches or regions from a huge pool and uses the responses of those discriminative elements as a mid-level representation for classification. However, this approach has mainly been applied to scene classification and not typically to fine-grained classification. This is probably due to the fact that the discriminative patches needed for fine-grained categories need to be more accurately localized than for scene classification.

To avoid the cost of extra annotations, we propose a *patch-based* approach for fine-grained classification that overcomes the difficulties of previous discriminative mid-level approaches. Our approach requires only object bounding box annotations and belongs to the category of weakly-annotated fine-grained classification [53–60].

Two issues need to be addressed. The first issue is accurately localizing discriminative patches without requiring extra annotations. Localizing a single patch based only on its appearance remains challenging due to noisy backgrounds, ambiguous repetitive patterns and pose variations. Instead, we localize *triplets* of patches with the help of geometric constraints. Previously, triple or higher-order constraints have been used in image matching and registration [61–63], but they have not been integrated into a patch-based classification framework. Our triplets consist of three appearance descriptors and two simple, but efficient, geometric constraints, which can be used to remove accidental detections that would be encountered if patches were localized individually. One attractive property of triplets over simpler models (*e.g.*, pairs) is that they can be used to model rich geometric relationships while providing additional invariance (*e.g.*, to rotation) or robustness (*e.g.*, to perspective changes).

The second issue is automatically discovering discriminative geometrically-constrained triplets from the huge pool of all possible triplets of patches. The key insight is that fine-grained objects share similar overall appearance. Therefore, if we retrieve the nearest neighbors of a training image, we obtain samples from different classes with almost the same pose, from which potentially discriminative regions can be found. Similar ideas have been adopted in [55, 64–66], but they aggregate

results obtained from local neighborhood processing without further analysis across the whole dataset. In contrast, our discriminative triplet mining framework uses sets of overall similar images to propose potential triplets, and only selects good ones by measuring their discriminativeness using the whole training set or a large portion of it.

We evaluate our approach on four publicly available fine-grained datasets and obtain comparable results to the state-of-the-art without expensive annotation.

3.2 Triplets of Patches with Geometric Constraints

In this section, we discuss two geometric constraints encoded within triplets of patches and describe a triplet detector incorporating these constraints.

Suppose we have three patch templates T_A, T_B and T_C, with their centers located at points A, B and C, respectively. Each template T_i ($i \in \{A, B, C\}$) can be a feature vector extracted from a single patch or an averaged feature vector of patches from the corresponding locations of several positive samples. Given an image, let A', B' and C' be three patches possibly corresponding to A, B and C, respectively.

3.2.1 Order Constraint and Shape Constraint

The order constraint encodes the ordering of the three patches (Figure 3.1). For triplet $\{A, B, C\}$, consider the two vectors $\overrightarrow{AB}$ and $\overrightarrow{AC}$. Treating them as three

dimensional vectors with the third dimension being 0, it follows that

$$\overrightarrow{AB} \times \overrightarrow{AC} = (0, 0, Z_{ABC}). \tag{3.1}$$

Let

$$G_{ABC} = \text{sign}(Z_{ABC}), \tag{3.2}$$

which indicates whether the three patches are arranged clockwise ($G_{ABC} = 1$) or counterclockwise ($G_{ABC} = -1$), as can be seen in Figure 3.1. There is a side-test interpretation of this constraint [67]. If we fix two patches, say B and C, $G_{ABC} = 1$ means that A lies on the left side of the line passing between B and C, while $G_{ABC} = -1$ indicates A is on the right side. Therefore, a simple penalty function between $\{A, B, C\}$ and $\{A', B', C'\}$ based on the order constraint can be defined as

$$p_o(G_{ABC}, G_{A'B'C'}) = 1 - \eta_o \mathbf{l}(G_{A'B'C'} \neq G_{ABC}), \tag{3.3}$$

where $0 \leq \eta_o \leq 1$ controls how important the order constraint is, and the indicator function is defined as

$$\mathbf{l}(G_{A'B'C'} \neq G_{ABC}) = \begin{cases} 1, & G_{A'B'C'} \neq G_{ABC} \\ 0, & G_{A'B'C'} = G_{ABC}. \end{cases} \tag{3.4}$$

Intuitively, p_o penalizes by η_o when $\{A', B', C'\}$ violates the order constraint defined by $\{A, B, C\}$.

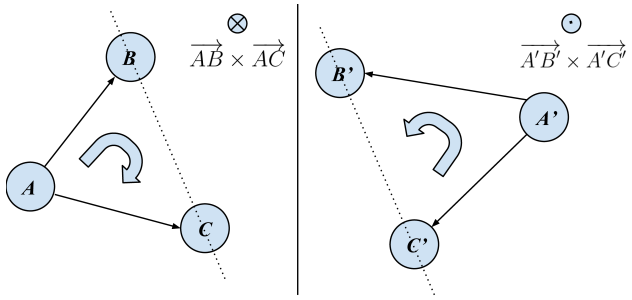

Figure 3.1: Visualization of the order constraint. Top: Patch A, B and C are arranged in clockwise order. The direction of $\overrightarrow{AB} \times \overrightarrow{AC}$ points into the paper, and $G_{ABC} = 1$. Bottom: A', B' and C' are arranged in counterclockwise order. The direction of the cross product points out of the paper, and $G_{A'B'C'} = -1$.

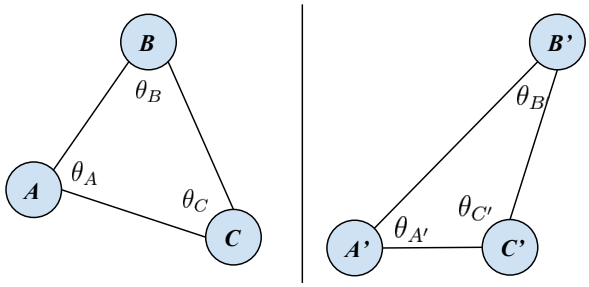

Figure 3.2: Visualization of the shape constraint. The constraint compares the three angles of two triplets.

The shape constraint measures the shape of the triangle defined by three patch centers (Figure 3.2). Let $\Theta_{ABC} = \{\theta_A, \theta_B, \theta_C\}$ denote the angles of triangle ABC, and $\Theta_{A'B'C'} = \{\theta_{A'}, \theta_{B'}, \theta_{C'}\}$ denote the angles of triangle $A'B'C'$, as displayed in Figure 3.2. We define a shape penalty function by comparing corresponding angles as

$$\begin{aligned} & p_s(\Theta_{A'B'C'}, \Theta_{ABC}) \\ =& 1 - \eta_s \frac{\sum_{i \in \{A,B,C\}} |\cos(\theta_i) - \cos(\theta_{i'})|}{6}, \end{aligned} \tag{3.5}$$

where $\eta_s \in [0, 1]$ controls how important the shape constraint is. The denominator 6 in Eq. (3.5) ensures that $0 \leq p_s \leq 1$, since $|\cos(\theta_i) - \cos(\theta_{i'})| \leq 2$. $\{\cos(\theta_i)\}$ and $\{\cos(\theta_{i'})\}$ can be easily computed from inner products. The motivation for introducing this second constraint is that we use the relatively loose order constraint to perform coarse verification and use the shape constraint for finer adjustments. As will be demonstrated in Section 3.4.1, the two constraints contain complementary information.

3.2.2 Triplet Detector

Our triplet detector consists of three appearance models and the two geometric constraints. We construct three linear weights $\{w_A, w_B, w_C\}$ from the patch templates $\{T_A, T_B, T_C\}$, and our triplet detector becomes

$$\mathbf{T} = \{w_A, w_B, w_C, G_{ABC}, \Theta_{ABC}\}. \tag{3.6}$$

For any triplet $\{A', B', C'\}$ with features $\{T_{A'}, T_{B'}, T_{C'}\}$ and geometric parameters $\{G_{A'B'C'}, \Theta_{A'B'C'}\}$, its detection score is defined as

$$S_{A'B'C'} = (S_A(T'_A) + S_B(T'_B) + S_C(T'_C)) \cdot p_o \cdot p_s, \tag{3.7}$$

where p_o, p_s are defined by Eq. (3.3), Eq. (3.5) respectively, and the appearance scores are defined as

$$S_i(T_{i'}) = w_i^{\mathrm{T}} T_{i'}, \quad i \in \{A, B, C\}. \tag{3.8}$$

To make triplet detection practical, three technical details must be addressed. The first is how to efficiently obtain the maximum response of a triplet detector in an image. In principle, we could examine all possible triplets from the combinations of all possible patches and simply compute

$$\{A^*, B^*, C^*\} = \underset{\{A', B', C'\}}{\operatorname{argmax}} S_{A'B'C'}. \tag{3.9}$$

However, this is too expensive since the number of all possible triplets will be $O(N^3)$ for N patches. Instead, we adopt a greedy approach. We first find the top K non-overlapping detections for each appearance detector independently. Then we evaluate the K^3 possible triplets and select the one with the maximum score defined by Eq. (3.7).

The second technical detail is how to obtain the linear weights w_i from a patch

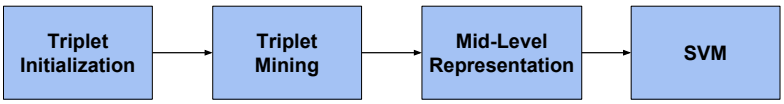

Figure 3.3: Summary of our discriminative triplet mining framework. Candidate triplets are initialized from sets of neighboring images and selected by how discriminative they are across the training set. The mid-level representation consists of the maximum responses of the selected triplets with geometric constraints, which is fed into a linear SVM for classification.

template T_i. For efficiency, we use the LDA model introduced by [68]

$$w_i = \Sigma^{-1} (T_i - \mu) , \tag{3.10}$$

where μ is the mean of features from all patches in the dataset, and Σ is the corresponding covariance matrix. The LDA model is efficient since it constructs the model for negative patches (μ and Σ) only once.

Our triplet detector is able to handle moderate pose variations. However, if we flip an image, both the appearance (*e.g.*the dominant direction of an edge) and the order of the patches will change. We address this issue by applying the triplet detector to the image and its mirror, generating two detection scores, and choose the larger score as the response. This simple technique proves effective in practice.

3.3 Discriminative Triplets for Fine-Grained Classification

In this section, we describe how to automatically mine discriminative triplets with the geometric constraints and generate mid-level representations for classification with the mined triplets. We present the overview of our framework in Fig-

ure 3.3. In the triplet initialization stage, we use a nearest-neighbor approach to propose potential triplets, taking advantage of the fact that instances of fine-grained objects share similar overall appearance. Then we verify the discriminativeness of the candidate triplets using the whole training set or a large portion of it, and select discriminative ones according to an entropy-based measure. For classification, we concatenate the maximum responses of the selected discriminative triplets to construct mid-level image representations. The key to our approach is proposing candidate triplets locally and selecting discriminative ones globally, avoiding the insufficient data problems of other nearest-neighbor based fine-grained approaches.

3.3.1 Triplet Initialization

To reduce the computational burden of triplet mining, we initialize candidate triplets using potentially discriminative patches in a nearest-neighbor fashion. The overview of the procedure is displayed in Figure 3.4.

Construct Neighborhood. For a seed training image I_0 with class label c_0, we extract features [69] of the whole image X_0 and use it to retrieve the nearest neighbors from the training set. Since fine-grained objects have similar overall appearance, the resulting set of images consists of training images from different classes with almost the same pose (Figure 3.5). This first step results in a set of roughly aligned images, so that potentially discriminative regions can be found by comparing corresponding regions across the images. We refer to the set consisting of a seed training image and its nearest neighbors as a neighborhood.

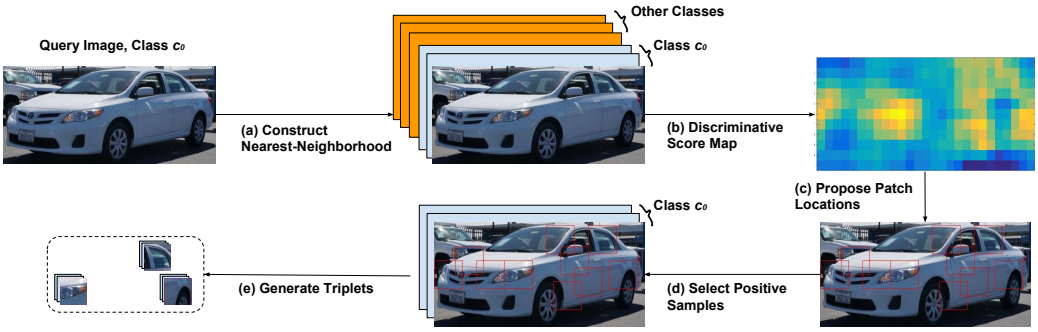

Figure 3.4: Visualization of the triplet initialization stage. **(a)** A seed image from class c_0 is used to construct its nearest-neighbor set including itself. **(b)** Discriminative score map is generated from the stack of neighboring images. **(c)** Patch locations with top discriminative scores are selected by non-maximum suppression. **(d)** Images from positive class (class c_0) are selected to generate triplets. **(e)** Triplets are generated from positive samples at locations proposed in (c).

Figure 3.5: Some examples of a set of neighboring images. The query image is highlighted with a red box. Since fine-grained objects share similar overall appearance, the neighborhood consists of samples from different classes with the same pose.

Find Candidate Regions. We regard each neighborhood as a stack of aligned images with their class labels and locate potentially discriminative regions. Consider the set of patches at the same location of each image. For each location (x, y), let $F_i(x, y)$ be the features extracted from the patch in the i^{th} image and let c_i be its label. Denote the set of observed class labels as C. The discriminative score at (x, y) is simply defined as the ratio of between-class variation and in-class variation, *i.e.*

$$d(x, y) = \frac{\sum_{c \in C} \left\| \overline{F_c}(x, y) - \overline{F}(x, y) \right\|^2}{\sum_{c \in C} \sum_{c_i = c} \left\| F_i(x, y) - \overline{F_c}(x, y) \right\|^2}, \tag{3.11}$$

where $\overline{F}(x, y)$ is the averaged feature of all patches at location (x, y), and $\overline{F_c}(x, y)$ is the average of patches from class c. We compute discriminative scores $d(x, y)$ for patch locations in a sliding window fashion, with patch size 64×64 and stride 8, and choose the patch locations with top scores. To ensure the diversity of the regions, non-maximum suppression is used to allow only a small amount of overlap.

Propose Candidate Triplets. For a neighborhood generated from the seed image with class label c_0, candidate triplets are proposed as follows. We first select all positive samples with label c_0 in the neighborhood. For each positive sample, we extract features from patches at the discriminative patch locations obtained in the last step. Then for patch location i, the patch template T_i is obtained by averaging the features of all the positive samples. We propose candidate triplets by selecting all possible combinations of three patch locations. To avoid duplicate triplets, we rank three patch locations by their discriminative scores Eq. (3.11). We construct triplet detectors with geometric constraints from these patch triplets as discussed

in Section 3.2.2.

In practice, in each neighborhood we find the top 6 discriminative locations and propose $\binom{6}{3} = 20$ candidate triplets. By considering all the neighborhoods for every class, we obtain the pool of candidate triplets.

3.3.2 Discriminative Triplets Mining by Entropy Scores

Candidate triplets are constructed from potentially discriminative regions measured by Eq. (3.11). However, this measure is computed only within a small neighborhood and might be noisy due to lack of data. On the other hand, recent approaches to scene understanding [48–51] mine discriminative patches using a large portion of the training data and obtain very good results. Consequently, we select discriminative triplets by evaluating each candidate triplet on the broader dataset.

For each triplet detector obtained in Section 3.3.1, we detect triplets in each training image and obtain the maximum detection score as discussed in Section 3.2.2, *i.e.*, find the top K detections for each appearance detector, consider all K^3 triplets, and choose the one with maximum geometrically-penalized score Eq. (3.7). We obtain the top detections within the training set along with their corresponding class labels. If a triplet is discriminative across the training set, the top detections are expected to arise from only one or a few classes. Therefore, if we calculate the entropy of the class distribution over the top detections, the entropy should be low. Let $p(c|\mathbf{T})$ denote the probability of top detections coming from class c for triplet

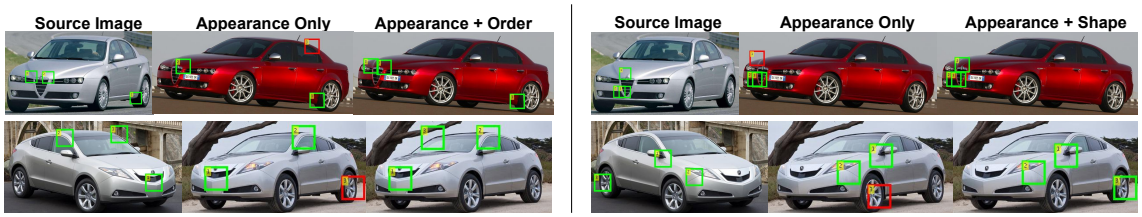

Figure 3.6: Visualization of the effects of the two geometric constraints. Red boxes are incorrectly localized patches. The order constraint roughly checks the geometric arrangement of three patches and can eliminate incidental false detections which happen to have high appearance score; the shape constraints enforce finer adjustments on patch locations than the order constraint.

$\mathbf{T}$. Then

$$H(c|\mathbf{T}) = \sum_{c} p(c|\mathbf{T}) \log p(c|\mathbf{T}) \tag{3.12}$$

is an entropy-based measure that has been effectively used by [51, 70] for patch selection. We calculate this measure for all candidate triplets and choose the ones with the lowest entropy to form the set of discriminative triplets.

3.3.3 Mid-Level Image Representations for Classification

Finally, we use the maximum responses of the mined discriminative triplet detectors to construct a mid-level representation for an image. The dimension of the mid-level representation equals the number of triplet detectors, with each detection score occupying one dimension in the image-level descriptor. The mid-level representation is used as the input for a linear SVM to produce the classification result. We refer to the image-level descriptor as a *Bag of Triplets* (BoT).

3.4 Experiments

3.4.1 Triplet Localization with Geometric Constraints

We first design a simple experiment to demonstrate that the geometric constraints described in Section 3.2 can actually improve patch localization, assuming we already have a good patch set. To achieve this goal, we use the FG3DCar dataset provided by [47]. This dataset consists of 300 car images from 30 classes, with each image annotated with 64 ground-truth landmark points. Cars in this dataset have large pose variations, which makes triplet localization difficult.

Experimental Settings. The experiment is designed as follows. For each image, we construct a good set of 64 patches by extracting the ones located at the annotated landmarks. We repeatedly and randomly select two images from the same class and obtain two corresponding sets of 64 patches with their locations. Then we randomly select three patches from one image (denoted as Image 1) to construct a triplet detector in Eq. (3.6) and attempt to find the corresponding triplet in the pool of 64 patches of the other (denoted as Image 2). During the experiment, patches are represented by the Fisher Vector features provided by [47].

The following four methods are evaluated on this task. The decision procedure of the three methods with gemetric constraints is discussed in Section 3.2.2.

- Appearance Only (Baseline): Independently apply each patch detector, and choose the detection with the highest score. The detected triplet consists of the three top individual detections. In this method, only the appearance features

of the three patches are used.

- Order Constraint: Use the appearance and the order constraint by setting $\eta_s = 0$ in Eq. (3.5), such that $p_s = 1$ (no shape penalty) always holds in Eq. (3.7).

- Shape Constraint: Use the appearance and the shape constraint by setting $\eta_o = 0$ in Eq. (3.3), such that $p_o = 1$ (no order penalty) always holds in Eq. (3.7).

- Combined: Use both geometric constraints. In practice, we set $\eta_o = 0.5$ and $\eta_s = 1$.

Due to large pose changes, several landmark locations are highly overlapped with each other in some images. Therefore, during evaluation, each patch is regarded as correctly localized if the detected patch is (i) the same as the ground truth corresponding patch; (ii) highly overlapped with the ground truth corresponding patch, with overlap/union ratio greater than 50%. Each triplet is regarded as successfully localized if all of its three patches are correctly localized. We randomly select 1000 image pairs, and for each pair we randomly test 1000 triplets. The accuracy of triplet localization is the percentage of successfully localized triplets over all the 1 million triplets evaluated.

Result and Analysis. We demonstrate triplet localization accuracy and relative improvement over baseline in Table 3.1. Even though we have a human-annotated pool of patches, localization is challenging with appearance only, since the pose vari-

Method	Localization Accuracy (%)	Improvement Over Baseline (%)
Appearance Only	24.9	-
Order Constraint	27.7	11.2
Shape Constraint	34.4	38.2
Combined	35.3	41.9

Table 3.1: Triplets localization test result on FG3DCar dataset. The localization accuracy and relative improvement over baseline (Appearance Only method) are demonstrated.

ations are large and the appearance detector is learnt with only one positive sample. As we add geometric constraints, we obtain cumulative improvement over the baseline. Typical examples indicating the effects of the two constraints are displayed in Figure 3.6. The order constraint, which is relatively loose, tends to roughly check the geometric arrangement of three patches. It can eliminate the patches which happen to have a very high appearance score. On the other hand, the shape constraint enforces fine adjustment, which is complementary to the order constraint. With the two geometric constraints combined, the improvement is significant.

In the following, we demonstrate fine-grained classification results on three standard fine-grained car datasets. No extra annotation beyond object bounding boxes is used throughout the experiments. When comparing results, we refer to our approach as *Bag of Triplets* (BoT).

3.4.2 14-Class BMVC Cars Dataset Results

Dataset. The fine-grained car dataset provided by [71] (denoted as BMVC-

14) consists of 1904 images of cars from 14 classes. [71] has split the data into 50% *train*, 25% *val* and 25% *test*. We follow this setting for evaluation.

Experimental Settings. The implementation details of our discriminative triplet mining approach are briefly stated as follows. Each image is cropped to its bounding box and resized such that the width is 500 (aspect ratio maintained). The patch size is set to be 64×64 and HOG features are extracted to represent the patches for fair comparison to preivously reported results. In the triplet initialization stage, for each seed image we construct the neighborhood of size 20 including itself. As mentioned in Section 3.3.1, for each neighborhood we propose the top 6 discriminative patch locations and propose $\binom{6}{3} = 20$ triplets for mining. In the triplet mining stage, we obtain the top detections across the whole training set and calculate entropy measure Eq. (3.12). Then we select 300 discriminative triplets per class. Finally, the mid-level representation has dimension $14 \times 300 = 4200$, which is fed into the linear SVM implemented by LIBLINEAR [72].

We test the following two cases:

- Without Geometric Constraints (Without Geo): In the discriminative mining and mid-level representation construction stages, we adopt the "Appearance Only" triplet detection strategy described in Section 3.4.1.

- With Geometric Constraints (With Geo): Each time we use a triplet detector, we use Eq. (3.7) and related techniques in Section 3.2.2 to incorporate the two constraints. By comparing the two cases, we quantitatively test the effectiveness of geometric constraints.

Method	Accuracy (%)
LLC [73]	84.5
PHOW [74]	89.0
FV [75]	93.9
structDPM [71]	93.5
BB-3D-G [36]	94.5
BoT (HOG Without Geo)	94.1
BoT (HOG With Geo)	96.6

Table 3.2: Results on BMVC-14 dataset.

Results and Analysis. We compare our results with previous work, citing the results from [47], which has provided a summary of previously published results on BMVC-14. It includes several baseline methods such as LLC [73] and PHOW [74] with codebook size 2048, Fisher Vector (FV) [75] with 256 Gaussian Mixture Model (GMM) components, as well as structDPM [71] and BB-3D-G [36] specifically designed for the task. Among these methods, BB-3D-G [36] used extra 3D models, while others only used ground truth bounding boxes as we did. The results are summarized in Table 3.2. Our method without geometric constraints outperforms all three baseline methods. When geometric constraints are further added, our approach not only outperforms the best reported result using only bounding boxes with a noticeable margin, but it also outperforms the BB-3D-G method which uses extra 3D model fitting. It is worth mentioning that our method with triplet geometric constraints outperforms the DPM-based method [71], which incorporates root-part pair-wise constraints.

We plot the performance as a function of the number of discriminative triplets

Figure 3.7: Visualization of the most discriminative triplet (measured by Eq. (3.12)) for each class in BMVC-14 proposed by our method. The triplets accurately capture the subtle discriminative information of each class, which is highly consistent with human perception. For instance, for the **first image in the first row**, the triplet captures the curvy nature of Volkswagen Beetle such as rounded hood; for the **first image in the second row**, the triplet focuses on the rear cargo of the pick-up truck, since Ford F-Series is the only pick-up in the dataset; for the **last image in the first row**, the triplet highlights the frontal face of Jeep Wrangler.

per class (with geometric constraints) in Figure 3.8. When we use only 10 triplets/class, the performance of 84.9% already outperforms the baseline LLC [73], suggesting that the mined discriminative triplets are highly informative. As we increase the number of triplets per class, performance more or less saturates after 100 triplets/class, although the best performance is at 300 triplets/class. Therefore, when we deal with large-scale datasets such as the Stanford Cars dataset below, we can use a smaller number of triplets to construct lower-dimensional mid-level descriptors without much loss in performance. As the number of triplets/class exceeds 300, the performance decreases, suggesting that the remaining triplets, which rank low by our criteria, do not add discriminatively useful information.

We further visualize the most discriminative triplet measured by the entropy score Eq. (3.12) for all 14 classes in Figure 3.7. The triplets in the figure accurately localize the subtle discriminative regions, which are highly consistent with human perception, such as the distinctive side vent grill of Chevrolet Corvette (the sec-

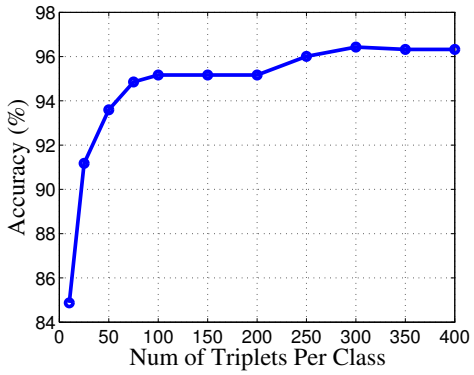

Figure 3.8: Classification accuracy with respect to the number of triplets per class.

ond image in the second row of Figure 3.7, see Figure 3.7 for more details). This empirically explains why our triplets are highly informative.

3.4.3 196-Class Stanford Cars Dataset Results

Dataset. The Stanford Cars Dataset [36] contains 16,185 car images from 196 classes (denoted as Cars-196). The data split provided by [36] is 8,144 training images and 8,041 testing images, where each class has been roughly split 50-50. We follow this setting in our experiment.

Experimental Settings. Our method focuses on generating effective mid-level representations and is independent from the choice of low-level features. In this experiment, in order to fairly compare to both the traditional methods without using extra data/annotation and the more recent ones which finetune ImageNet pre-trained Convolutional Neural Networks (CNN), we evaluate our approach using both HOG and CNN features as the low-level representations of the patches. When extracting CNN features, we directly use the off-the-shelf ImageNet pre-trained CNN

model as a general feature extractor without any finetuning. For fair comparison, we adopt the popular 16-layer VGGNet-16 [8] as the network architecture, and extract features from pool_4 layer, which is the max-pooled output of its 10^{th} convolutional layer.

Also, we adapt our approach slightly to handle such a large-scale dataset. Instead of traversing the whole dataset, when retrieving nearest-neighbors for a seed image with class label c_0, we regard class c_0 as the positive class, randomly select 29 other classes as negative classes, and retrieve nearest-neighbors within the training images from these 30 classes; when finding top detections for a triplet from class c_0, we use the training images from class c_0 and 14 randomly selected negative classes. Additionally, since we have empirically determined that the discriminative triplets are informative in Figure 3.8, we select 150 discriminative triplets per class.

Except for the settings described above, other parameters remain the same as those in Section 3.4.2.

Results and Analysis. Our baselines include LLC [73] as HOG-based baseline and AlexNet [76] as CNN baselines. For CNN, we cite the result of training an AlexNet from scratch on Cars-196 without extra data (AlexNet From Scratch) [66] and the result of finetuning an ImageNet pre-trained AlexNet on Cars-196 (AlexNet Finetuned) [58]. We also compare with previously published results including BB-3D-G [36], ELLF [66], FT-HAR-CNN [58], Bilinear CNN (B-CNN) [59] and the method with the highest reported accuracy so far [60]. It is worth mentioning that the last three approaches [58–60] are CNN-based, where [58] is AlexNet based, [59] is VGGNet-16 based, and [60] is 19-layer VGGNet-19 based. [58] finetunes a CNN

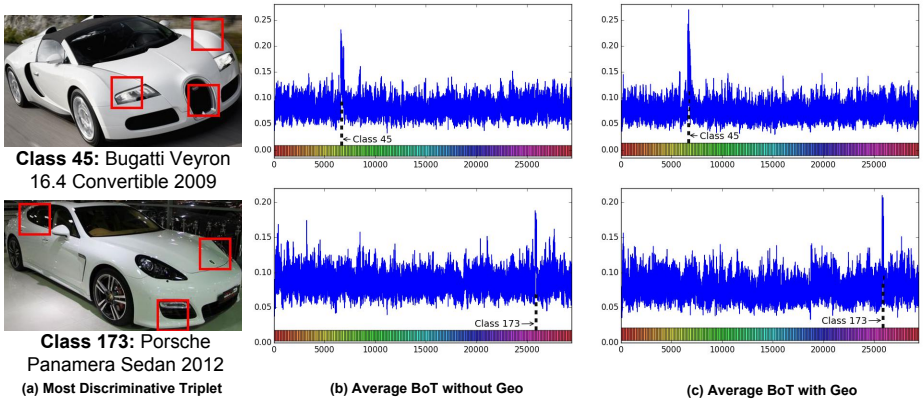

Figure 3.9: (a) Visualization of the most discriminative triplets of two example classes in Cars-196. (b)(c) The averaged image-level BoT descriptor across all test samples in the corresponding class. Each dimension in the BoT is generated by the response of a mined triplet. The color bars are used to describe the dimensions correponding to the responses of triplets from different classes.

with the help of another 10,000 images of cars without fine-grained labels; the best result of [59] is achieved by finetuning a two-stream CNN architecture; [60] integrates segmentation, graph-based alignment, finetuned R-CNN [77] and SVM to produce its best result.

The results are displayed in Table 3.3. Even though we adopted relatively "economical" settings, our method behaves stably and operates at the state-of-the-art performance. When using HOG as low-level patch representation, our approach not only greatly outperforms the HOG-based baseline (LLC) (by more than 15%), but it even outperforms the CNN baseline of finetuned AlexNet by a fairly noticeable margin (more than 2%) – a significant achievement since we are only using HOG and geometric constraints without any extra data or annotations. When using off-the-shelf CNN features, our method with geometric constraints outperforms B-CNN [59] which uses two streams of VGGNet-16, and obtained quite comparable results to

the state-of-the-art [60]. Furthermore, our method does not perform finetuning and depends on the strength of our discriminative triplet mining itself, which is supported by the results, rather than the learning capability of CNNs.

To intuitively demonstrate the effectiveness of the geometric constraints, we plot the image-level BoT descriptors from a few classes in the second and third columns of Figure 3.9. For each class we plot the averaged BoT descriptor across all *test* samples from that class. Figure 3.9 shows that after introducing the geometric constraints, the BoT descriptor becomes more peaked at the corresponding class, since the geometric constraints help learn more discriminative triplets which generate more peaky responses, as well as penalizing those incorrect detections from other classes which happen to have high appearance scores (which can be clearly seen from the second row of Figure 3.9). This discriminative capability is achieved during test time, showing that our approach generalizes very well.

Finally, we visualize the most discriminative triplet measured by Eq. (3.12) in the first column of Figure 3.9. Similar to Figure 3.7, our approach captures the subtle difference of fine-grained categories and accurately localizes the discriminative regions, which are highly interpretable by humans. For example, it highlights the distinctive air grill and rounded fender of Bugatti Veyron, and the classical headlight and tail of Porsche.

Method	Accuracy (%)
LLC* [73]	69.5
BB-3D-G [36]	67.6
ELLF* [66]	73.9
AlexNet From Scratch [66]	70.5
AlexNet Finetuned [58]	83.1
FT-HAR-CNN [58]	86.3
B-CNN [59]	91.3
Best Result in [60]	92.8
BoT(HOG Without Geo)*	84.6
BoT(HOG With Geo)*	85.7
BoT(CNN Without Geo)	91.2
BoT(CNN With Geo)	92.5

Table 3.3: Results on Cars-196 dataset. Items with "*" indicate that no extra annotations/data are involved.

Method	Accuracy (%)
Symbiotic [57]	75.9
Fine-tuned AlexNet [78]	78.9
Fisher Vector [78]	81.5
B-CNN [59]	84.1
BoT (CNN without Geo)	86.7
BoT (CNN with Geo)	88.4

Table 3.4: Results on FGVC-Aircraft dataset.

3.4.4 100-Class FGVC-Aircraft Dataset Results

Finally, to demonstrate that our approach is effective in multiple fine-grained domains, we briefly present our results on FGVC-Aircraft dataset [79], which contains 10,000 images from 100 classes of aircrafts and is of similar scale to the Cars-196 dataset (16185 images from 196 classes). For fair comparison, we use the standard train/test split provided by the dataset provider [79] and the parameter settings of our approach are *exactly the same* as those in Section 3.4.3. We report our results in Table 3.4. Our approach using CNN features (without fine-tuning) outperforms state-of-the-art (VGGNet-16 based) [59] by a noticeable margin. The results suggest that our approach performs well in various fine-grained domains.

Chapter 4: Learning Discriminative Features via Label Consistent Neural Network

4.1 Background and Motivation

Convolutional neural networks (CNN) [80] have exhibited impressive performances in many computer vision tasks such as image classification [9], object detection [77] and image retrieval [81]. One key reason is the availability of a large amount of training data. They can automatically learn hierarchical feature representations which are more discriminative than previous hand-crafted ones [9].

Encouraged by their performance in static image analysis tasks, several CNN-based approaches have been developed for action recognition in videos [3, 82–86]. Although promising results have been reported, the advantages of CNN approaches over traditional ones [87] are not as overwhelming for videos as in static images. Compared to static images, videos have larger variations in appearance as well as high complexity introduced by temporal evolution, which makes learning features for recognition from videos more challenging. On the other hand, unlike large-scale and diverse static image data [88], annotated data for action recognition tasks is usually insufficient, since annotating massive videos is prohibitively expensive.

Therefore, with only limited annotated data, learning discriminative features via deep neural network can lead to severe overfitting and slow convergence. To tackle these issues, previous works have introduced effective practical techniques such as ReLU [89] and Drop-out [90] to improve the performance of neural networks, but have not considered directly improving the discriminative capability of neurons. The features from a CNN are learned by back-propagating prediction error from the output layer [91], and hidden layers receive no direct guidance on class information. Worse, in very deep networks, the early hidden layers often suffer from vanishing gradients, which leads to slow optimization convergence and the network converging to a poor local minimum. Therefore, the quality of the learned features of the hidden layers might be potentially lowered [92, 93].

To tackle these problems, we propose a new supervised deep neural network, *Label Consistent Neural Network*, to learn discriminative features for recognition. Our approach provides explicit supervision, *i.e.* label information, to late hidden layers, by incorporating a label consistency constraint called "discriminative representation error" loss, which is combined with the classification loss to form the overall objective function. The benefits of our approach are two-fold: (1) with explicit supervision to hidden layers, the problem of vanishing gradients can be alleviated and faster convergence is observed; (2) more discriminative late hidden layer features lead to increasing discriminative power of classifiers at the output layer; interestingly, the learned discriminative features alone can achieve good classification performance even with a simple k-NN classifier. In practice, our new formulation can be easily incorporated into any neural network trained using backpropagation. Our

approach is evaluated on publicly available action and object recognition datasets. Although we only present experimental results for action and object recognition, the method can be applied to other tasks such as image retrieval, compression, restorations *etc.*, since it generates class-specific compact representations.

The main contributions of LCNN are three-fold.

- By adding explicit supervision to late hidden layers via a "discriminative representation error", LCNN learns more discriminative features resulting in better classifier training at the output layer. The representations generated by late hidden layers are discriminative enough to achieve good performance using a simple k-NN classifier.

- The label consistency constraint alleviates the problem of vanishing gradients and leads to faster convergence during training, especially when limited training data is available.

- We achieve state-of-the-art performance on several action and object category recognition tasks, and the compact class-specific representations generated by LCNN can be directly used in other applications.

4.2 Feature Learning via Supervised Deep Learning

Let $(\mathbf{x}, y)$ denote a training sample $\mathbf{x}$ and its label y. For a CNN with n layers, let $\mathbf{x}^{(i)}$ denote the output of the i^{th} layer and L_c its objective function. $\mathbf{x}^{(0)} = \mathbf{x}$ is the input data and $\mathbf{x}^{(n)}$ is the output of the network. Therefore, the network

architecture can be concisely expressed as

$$\mathbf{x}^{(i)} = F(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)}), \quad i = 1, 2, ..., n \tag{4.1}$$

$$L_c = L_c(\mathbf{x}, y, \mathbf{W}) = C(\mathbf{x}^{(n)}, y), \tag{4.2}$$

where $\mathbf{W}^{(i)}$ represents the network parameters of the i^{th} layer, $\mathbf{W}^{(i)}\mathbf{x}^{(i-1)}$ is the linear operation (*e.g.*convolution in convolutional layer, or linear transformation in fully-connected layer), and $\mathbf{W} = \{\mathbf{W}^{(i)}\}_{i=1,2,...,n}$; $F(\cdot)$ is a non-linear activation function (*e.g.*ReLU); $C(\cdot)$ is some prediction error such as softmax loss. The network is trained with back-propagation, and the gradients are computed as

$$\frac{\partial L_c}{\partial \mathbf{x}^{(i)}} = \begin{cases} \frac{\partial C(\mathbf{x}^{(n)}, y)}{\partial \mathbf{x}^{(n)}}, & i = n \\ \frac{\partial L_c}{\partial \mathbf{x}^{(i+1)}} \frac{\partial F(\mathbf{W}^{(i+1)}\mathbf{x}^{(i)})}{\partial \mathbf{x}^{(i)}}, & i \neq n \end{cases} \tag{4.3}$$

$$\frac{\partial L_c}{\partial \mathbf{W}^{(i)}} = \frac{\partial L_c}{\partial \mathbf{x}^{(i)}} \frac{\partial F(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)})}{\partial \mathbf{W}^{(i)}}, \tag{4.4}$$

where $i = 1, 2, 3, ..., n$.

4.3 Label Consistent Neural Network (LCNN)

4.3.1 Motivation

The sparse representation for classification assumes that a testing sample can be well represented by training samples from the same class [94]. Similarly, dictionary learning for recognition maintains the label information for dictionary

items during training in order to generate discriminative or class-specific sparse codes [95,96]. In a neural network, the representation of a certain layer is generated by the neuron activations in that layer. If the class distribution for each neuron is highly peaked in one class, it enforces a label consistency constraint on each neuron. This leads to discriminative representation over learned class-specific neurons.

It has been observed that early hidden layers of a CNN tend to capture low-level features shared across categories such as edges and corners, while late hidden layers are more class-specific [92]. To improve the discriminativeness of features, LCNN adds explicit supervision to late hidden layers; more specifically, we associate each neuron to a certain class label and ideally the neuron will only activate when a sample of the corresponding class is presented. The label consistency constraint on neurons in LCNN will be imposed by introducing a "discriminative representation error" loss on late hidden layers, which will form part of the objective function during training.

4.3.2 Formulation

Specifically, the overall objective function of LCNN is a combination of the discriminative representation error at late hidden layers and the classification error at the output layer:

$$L = L_c + \alpha L_r \tag{4.5}$$

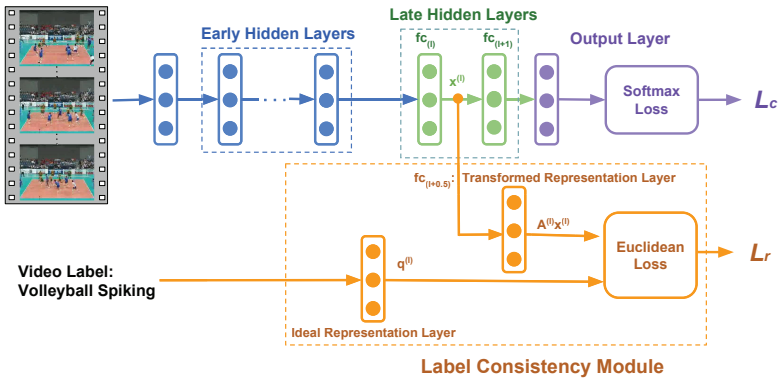

Figure 4.1: An example of the structure of LCNN. The label consistency module is added to the first late hidden layer, which is a fully-connected layer fc_l. Its representation $\mathbf{x}^l$ is transformed to be $\mathbf{A}^{(l)}\mathbf{x}^l$, which is the output of the transformed representation layer $\text{fc}_{l+0.5}$. Note that the applicability of the proposed label consistency module is not limited to fully-connected layers.

where L_c in Equation (4.2) is the classification error at the output layer, L_r is the discriminative representation error in Equation (4.6) and will be discussed in detail below, and α is a hyper parameter balancing the two terms.

Suppose we want to add supervision to the l^{th} layer. Let $(\mathbf{x}, y)$ denote a training sample and $\mathbf{x}^{(l)} \in \mathbb{R}^{N_l}$ be the corresponding representation produced by the l^{th} layer, which depends on the activations of N_l neurons in that layer. Then the discriminative representation error is defined to be the difference between the transformed representation $\mathbf{A}^{(l)}\mathbf{x}^{(l)}$ and the ideal discriminative representation $\mathbf{q}^{(l)}$:

$$L_r = L_r(\mathbf{x}^{(l)}, y, \mathbf{A}^{(l)}) = \|\mathbf{q}^{(l)} - \mathbf{A}^{(l)}\mathbf{x}^{(l)}\|_2^2, \tag{4.6}$$

where $\mathbf{A}^{(l)} \in \mathbb{R}^{N_l \times N_l}$ is a linear transformation matrix, and the binary vector $\mathbf{q}^{(l)} = [q_1^{(l)}, \ldots, q_j^{(l)}, \ldots, q_{N_l}^{(l)}]^{\mathrm{T}} \in \{0, 1\}^{N_l}$ denotes the ideal discriminative representa-

tion which indicates the ideal activations of neurons (j denotes the index of neuron, *i.e.*the index of feature dimension). Each neuron is associated with a certain class label and, ideally, only activates to samples from that class. Therefore, when a sample is from Class c, $q_j^{(l)} = 1$ if and only if the j^{th} neuron is assigned to Class c, and neurons associated to other classes should not be activated so that the corresponding entry in $\mathbf{q}^{(l)}$ is zero. Notice that $\mathbf{A}^{(l)}$ is the only parameter needed to be learned, while $\mathbf{q}^{(l)}$ is pre-defined based on label information from training data.

Suppose we have a batch of five training samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_5\}$ and the class labels $\mathbf{y} = [y_1, y_2, \ldots, y_5] = [1, 2, 2, 3, 3]$. Further assume that the l^{th} layer has 6 neurons $\{d_1, d_2, \ldots, d_6\}$ with $\{d_1\}$ associated with Class 1, $\{d_2, d_3, d_4\}$ Class 2, and $\{d_5, d_6\}$ Class 3. Then the ideal discriminative representations for these five samples are given by

$$\mathbf{Q}^{(l)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \tag{4.7}$$

where each column is an ideal discriminative representation corresponding to an input signal or sample. The ideal representations ensured that the input signals from the same class have similar representations while those from different classes

have dissimilar representations.

The discriminative representation error (4.6) forces the learned representation to approximate the ideal discriminative representation, so that the resulting neurons have the label consistency property [95], *i.e.*the class distributions of each neuron [1] from layer l are extremely peaked in one class. In addition, with more discriminative representations, the classifier, especially linear classifiers, at the output layer can achieve better performance. This is because the discriminative property of $\mathbf{x}^{(l)}$ is very important for the performance of a linear classifier.

An example of the implementation of LCNN is shown in Figure 4.1. The linear transformation is implemented as a fully-connected layer. We refer it as 'Transformed Representation Layer'. We create a new 'Ideal Representation Layer' which transforms a class label into the corresponding binary vector $\mathbf{q}^{(l)}$; then we feed the outputs of these two layers into a Euclidean loss layer. In our experiments, we uniformly allocate the neurons in the late hidden layer to each category[2]. So each neuron in the late hidden layer can be associated with a label. As the representations shown in Figure 4.4(i), an input signal of a category certainly can (and does) use neurons (learned features) from other categories, indicating that sharing feature between categories is not stopped.

[1] Similar to computing the class distributions for dictionary items in [97], the class distributions of each neurons from the l^{th} layer can be derived by measuring their activations $\mathbf{x}^{(l)}$ over input signals corresponding to different classes.

[2] Assuming N_l neurons in the layer and m classes, we allocate $[N_l/m]$ neurons to each class and then allocate remaining neurons to those classes with high intra-class difference.

4.3.3 Network Training

LCNN is trained via stochastic gradient descent. Now we need to compute the gradients of L in Equation (4.5) w.r.t. all the network parameters $\{\mathbf{W}, \mathbf{A}^{(l)}\}$. Compared with standard CNN, the difference lies in two gradient terms, *i.e.* $\frac{\partial L}{\partial \mathbf{x}^{(l)}}$ and $\frac{\partial L}{\partial \mathbf{A}^{(l)}}$, since $\mathbf{x}^{(l)}$ and $\mathbf{A}^{(l)}$ are the only parameters which are related to the newly added discriminative error $L_r(\mathbf{x}^{(l)}, y, \mathbf{A}^{(l)})$ and the other parameters act independently from it.

It follows from Equation (4.5) and (4.6) that

$$\frac{\partial L}{\partial \mathbf{x}^{(i)}} = \begin{cases} \frac{\partial L_c}{\partial \mathbf{x}^{(i)}}, & i \neq l \\ \frac{\partial L_c}{\partial \mathbf{x}^{(l)}} + 2\alpha(\mathbf{A}^{(l)}\mathbf{x}^{(l)} - \mathbf{q}^{(l)})^{\mathrm{T}}\mathbf{A}^{(l)}, & i = l \end{cases} \tag{4.8}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(i)}} = \frac{\partial L_c}{\partial \mathbf{W}^{(i)}}, \quad \forall i \in \{1, 2, ..., n\} \tag{4.9}$$

$$\frac{\partial L}{\partial \mathbf{A}^{(l)}} = 2\alpha(\mathbf{A}^{(l)}\mathbf{x}^{(l)} - \mathbf{q}^{(l)})\mathbf{x}^{(l)\mathrm{T}}, \tag{4.10}$$

where $\frac{\partial L_c}{\partial \mathbf{x}^{(i)}}$ and $\frac{\partial L_c}{\partial \mathbf{W}^{(i)}}$ are computed by Equation (4.3) and (4.4), respectively.

4.4 Experiments

We evaluate our approach on two action recognition datasets: UCF101 [98] and THUMOS15 [7], and three object category datasets: Cifar-10 [99], ImageNet [88] and Caltech101 [100]. Our implementation of LCNN is based on the CAFFE [101] toolbox.

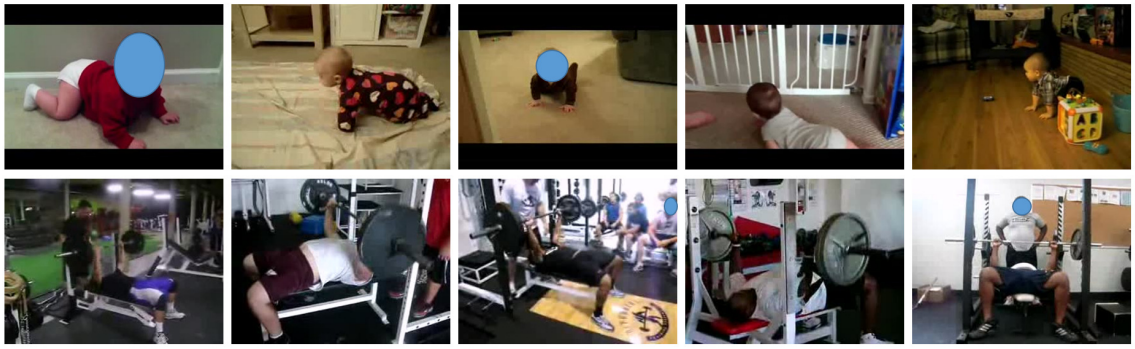

Figure 4.2: Class 4 (BabyCrawling) and class 10 (BenchPress) samples from the UCF101 action dataset.

During training, the objective function of LCNN in (4.5) is the combination of softmax classification error loss L_c and the discriminative representation error loss L_r. We refer to it as LCNN-2. Also, in order to verify the effect of our label consistency constraint, we use L_r only to train the network, which we refer to it as LCNN-1, in the following.

For action and object recognition, we introduce two classification approaches here: (1) we follow the standard CNN practice of taking the class label corresponding to the maximum prediction score; (2) We use the transformed representation $\mathbf{A}^{(l)}\mathbf{x}^{(l)}$ to represent an image, video frame or optical flow field and then use a simple k-NN classifier. We refer to these two approaches as 'argmax' and 'k-NN', respectively in the following.

4.4.1 Action Recognition

UCF101 Dataset The UCF101 dataset [98] consists of $13,320$ video clips from 101 action classes, and every class has more than 100 clips. Some video examples

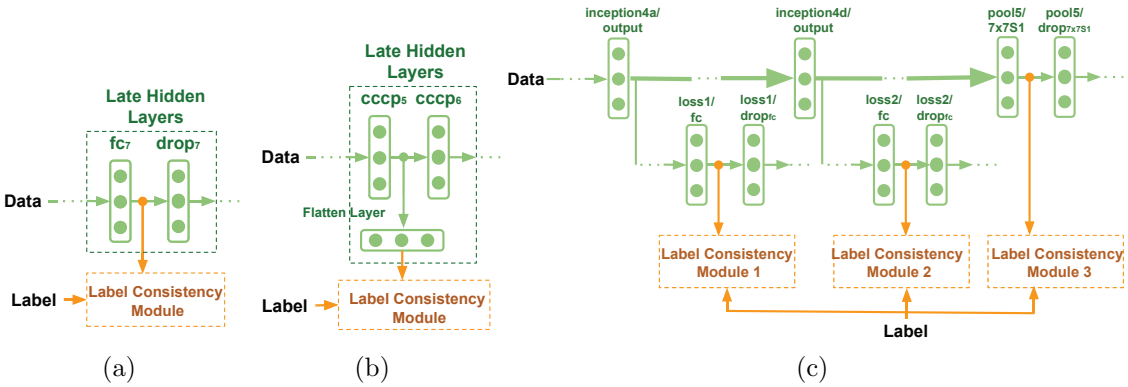

Figure 4.3: Examples of direction supervision in the late hidden layers including (a) fc_7 layer in the CNN architectures including VGG [8] and AlexNet [9]; (b) CCCP5 layer in the Network-in-Network [10];(c) $loss_1/fc$, $loss_2/fc$ and $Pool_5/7 \times 7S_1$ in the GoogLeNet [5]. The symbol of three dots denotes other layers in the network.

from class 4 and class 10 are given in Figure 4.2. In terms of evaluation, we use the standard split-1 train/test setting to evaluate our approach. Split-1 contains around $10,000$ clips for training and the rest for testing.

We choose the popular two-stream CNN as in [2–4] as our basic network architecture for action recognition. It consists of a spatial net taking video frames as input and a temporal net taking 10-frame stacking of optical flow fields. Late fusion is conducted on the outputs of the two streams and generates the final prediction score. During testing, we sample 25 frames (images or optical flow fields) from a video as in [3] for spatial and temporal nets. The class scores for a testing video is obtained by averaging the scores across sampled frames. In our experiments, we fuse spatial and temporal net prediction scores using a simple weighted average rule, where the weight is set as 2 for temporal net and 1 for spatial net.

For spatial and temporal nets, we use the VGGNet-16 architecture [8] as in [4]

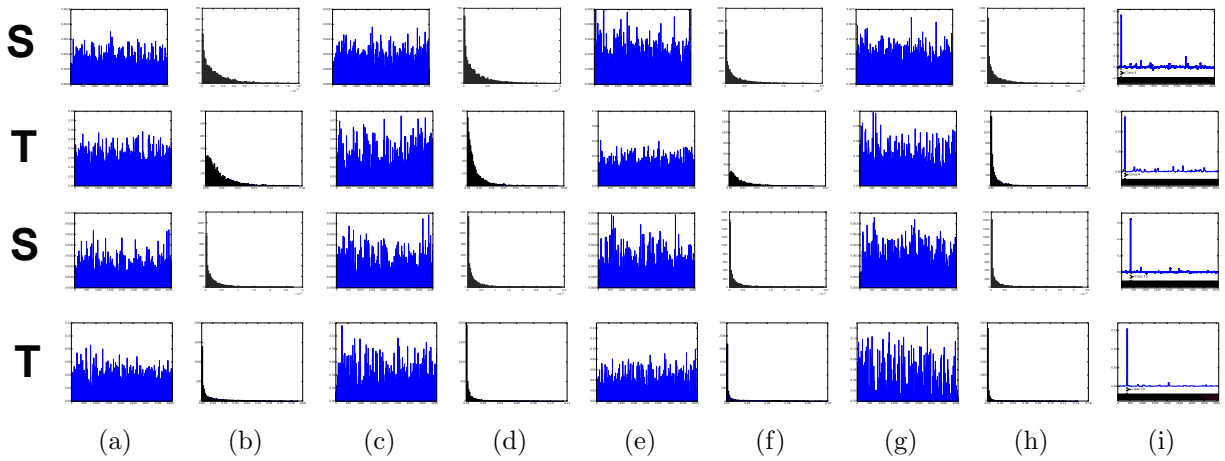

Figure 4.4: Examples of learned representations from layers fc_6, fc_7 and $\text{fc}_{7.5}$ using LCNN and VGGNet-16. Each curve indicates an average of representations for different testing videos from the same class in the UCF101 dataset. The first two rows correspond to class 4 (Baby Crawling, 35 videos) while the third and fourth rows correspond to class 10 (Bench Press, 48 videos). The curves in every two rows correspond to the spatial net (denoted as 'S') and temporal net (denoted as 'T') in our two-stream framework for action recognition. (a) fc_6 representations using VGGNet-16; (b) Histograms (with 100 bins) for representations from (a); (c) fc_6 representations using LCNN; (d) Histograms for representations from (c); (e) fc_7 representations using VGGNet-16; (f) Histograms for representations from (e); (g) fc_7 representations using LCNN; (h) Histograms for representations from (g); (i) $\text{fc}_{7.5}$ representations (i.e. transformed fc_7 representations) using LCNN. The entropy values for representations from (a)(c)(e)(g) are computed as: (11.32, 11.42, 11.02, 10.75), (11.2, 11.14, 10.81, 10.34), (11.08, 11.35, 10.67, 10.17), (11.02, 10.72, 10.55, 9.37). LCNN can generate lower-entropy representations for each class compared to VGGNet-16. The figure is best viewed in color and 600% zoom in.

for two streams where the explicit supervision is added in the late hidden layer fc_7, which is the second fully-connected layer. More specifically, we feed the output of layer fc_7 to a fully-connected layer (denoted as $\text{fc}_{7.5}$) to produce the transformed representation, and compare it to the ideal discriminative representation $\mathbf{q}^{(\text{fc7})}$. The implementation of this explicit supervision is shown in Figure 4.3(a). Since UCF101 has 101 classes and the fc_7 layer of VGGNet has output dimension 4096, the output of $\text{fc}_{7.5}$ has the same size 4096, and around 40 neurons are associated to each class. For both streams, we set $\alpha = 0.05$ in (4.5) to balance the two loss terms.

Network Architecture	Spatial	Temporal	Both
ClarifaiNet [3]	72.7	81	87
VGGNet-19 [4]	75.7	78.3	86.7
VGGNet-16 [2]	79.8	85.7	90.9
VGGNet-16* [2]	-	85.2	-
VGGNet-16** [2](baseline)	77.48	83.71	-
LCNN-1 (k-NN)	80.1	85.59	89.87
LCNN-2 (argmax)	80.7	85.57	91.12
LCNN-2 (k-NN)	81.3	85.77	89.84

Table 4.1: Classification performances with different two-stream CNN approaches on the UCF101 dataset (split-1). The results of [2], [3] and [4] are copied from their original paper. The VGGNet-16* result is obtained by running the original model trained and shared by [2], while VGGNet-16** is the reproduced result by using the same parameters and initial model provided by [2].

Method	Acc. (%)	Method	Acc. (%)
Karpathy [83]	65.4	Wang [87]	85.9
Donahue [102]	82.9	Lan [6]	89.1
Ng [84]	88.6	Zha [86]	89.6
LCNN-2 (argmax)	91.12		

Table 4.2: Recognition performance comparisons with state-of-the-art approaches on the UCF101 dataset.

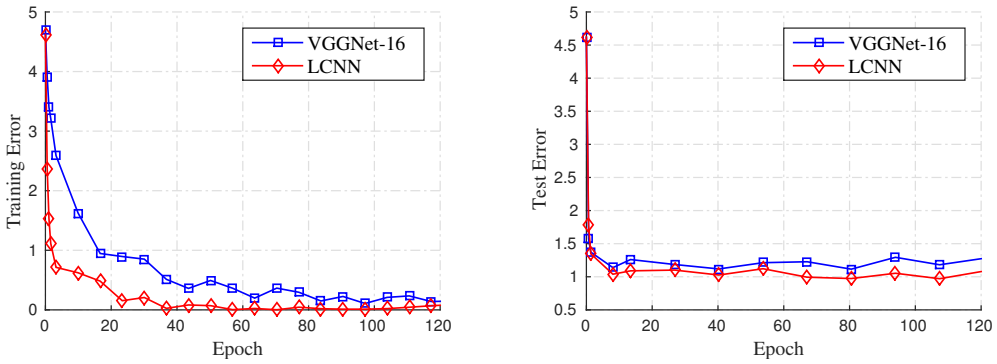

Figure 4.5: Training and testing error comparisons (based on spatial net) between LCNN-2 and VGGNet-16 on the UCF101 dataset. (a) Training error comparison; (b) Testing error comparison.

The results are summarized in Table 4.1. Under the same settings, by comparing the results of VGGNet-16** [2] and our LCNN-2, we can see that adding explicit supervision to late hidden layers not only improves the classification results at the output layer (LCNN-2 (argmax)), but also generates discriminative representations which achieve better results even with a simple k-NN classifier (LCNN-2 (k-NN)). Moreover, it can be seen from the results of LCNN-1 that even without the help of the classifier, our label consistency constraint alone is very effective for learning discriminative features and achieving good classification performance. We also compare our LCNN with other state-of-the-art approaches in Table 4.2.

In addition, we visualize the representations of *test videos* generated by late hidden layers $\text{fc}_{7.5}$, fc_7 and fc_6 in Figure 4.4. It can be seen that the entries of layer $\text{fc}_{7.5}$ representations are very peaked at the corresponding class and approaching zero elsewhere, which forms a very good approximation to the ideal discriminative representation. Further notice that such discriminative capability is achieved

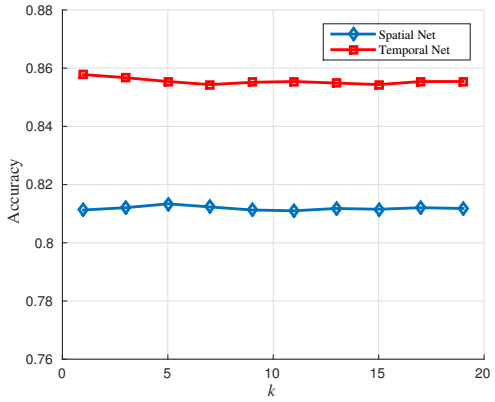

Figure 4.6: Effects of parameter selection of k-NN neighborhood size k on the classification accuracy performances on the UCF101 dataset.

during testing, which indicates that LCNN generalizes well without severe overfitting. For fc_7 and fc_6 representations, their entropy has decreased, which means that the discriminativeness of previous layers benefits from the backpropagation of the discriminative representation error introduced by LCNN.

We investigate the convergence and testing error of LCNN during network training. We plot the testing error and training error w.r.t. number of epochs in Figure 4.5. It can be seen that LCNN has smaller training error, which can converge more quickly and alleviate gradient vanishing due to the explicit supervision to late hidden layers. In addition, LCNN has smaller testing error compared with VGG, which means that LCNN has better generalization capability.

In Figure 4.6, we plot the performance curves for a range of k (recall k is the number of nearest neighbors for a k-NN classifier) using LCNN-2. We observe that our approach is insensitive to the selection of k, likely due to the increase of inter-class distances in generated class-specific representations.

THUMOS15 Dataset Next we evaluate our approach on the more challenging THUMOS15 challege action dataset. It includes all 13,320 video clips from UCF101 dataset for training, and $2,104$ temporarily *untrimmed* videos from the 101 classes for validation. We employ the standard Mean Average Precision (mAP) for THUMOS15 recognition task to evaluate our recognition performance.

We use the VGGNet-16 discussed in Section 4.4.1 as the underlying architecture and train it using all UCF101 data. We used the evaluation tool provided by the dataset provider to evaluate classification performance, which requires the probabilities for each category for a testing video. For our two classification schemes, *i.e.*argmax and k-NN, we use different approaches to generate the probability prediction for a testing video. For argmax, we can directly use the output layer. For the k-NN scheme, given the representation from $\text{fc}_{7.5}$ layer, we compute a sample's distances to classes only presented in its k nearest neighbors, and convert them to similarity weights using a Gaussian kernel and set other classes to be very low similarity; finally we calculate the probability by doing L1 normalization on the similarity vector.

The results are summarized in Table 4.3. Our results in the spatial stream outperform the results in [2], [3] and [5], while our results in the temporal stream are comparable to [3]. Based on this experiment, we can see that LCNN is highly effective and generalizes well to more complex testing data.

4.4.2 Object Recognition

Network Architecture	Spatial	Temporal	Both
VGGNet-16 [2]	54.5	42.6	-
ClarifaiNet [3]	42.3	47	-
GoogLeNet [5]	53.7	39.9	-
VGGNet-16* (baseline)	55.8	41.8	-
LCNN-1 (k-NN)	56.9	45.1	59.8
LCNN-2 (argmax)	57.3	44.9	61.7
LCNN-2 (k-NN)	58.6	45.9	62.6

Table 4.3: Mean Average Precision performance on the THUMOS15 validation set. The results of [2], [3] and [5] are copied from [2]. VGGNet-16* is the result of using the softmax loss only to train the network. Our result 62.6% mAP is also better than 54.7% using method in [6], which is reported in [7].

CIFAR-10 Dataset The CIFAR-10 dataset contains $60,000$ color images from 10 classes, which are split into $50,000$ training images and 10,000 testing images. We compare LCNN-2 with several recently proposed techniques, especially the Deeply Supervised Net (DSN) [103], which adds explicit supervision to all hidden layers. For our underlying architecture, we also choose Network in Network (NIN) [10] as in [103]. We follow the same data augmentation techniques in [10] by zero padding on each side, then do corner cropping and random flipping during training.

For our LCNN, we add the explicit supervision to the 5^{th} cascaded cross channel parametric pooling layer (cccp_5) [10], which is a late 1×1 convolutional layer. We first flatten the output of this convolutional layer into a one dimensional vector, and then feed it into a fully-connected layer (denoted as $\text{fc}_{5.5}$) to obtain the transformed representation. This implementation is shown in Figure 4.3(b). We set the hyper-parameter $\alpha = 0.0375$ during training. For classification, we adopt the argmax classification scheme.

Method (Without Data Augment.)	Test Error (%)
Stochastic Pooling [104]	15.13
Maxout Networks [105]	11.68
DSN [103]	9.78
NIN [10] (baseline)	10.41
LCNN-2 (argmax)	9.75
Method (With Data Augment.)	**Test Error (%)**
Maxout Networks [105]	9.38
DropConnect [106]	9.32
DSN [103]	8.22
NIN [10] (baseline)	8.81
LCNN-2 (argmax)	8.14

Table 4.4: Test error rates from different approaches on the CIFAR-10 dataset.

The results are summarized in Table 4.4. Regardless of the data augmentation, LCNN-2 consistently outperforms all previous methods, including NIN [10] and DSN [103]. The results are impressive, since DSN adds an SVM loss to every hidden layer during training, while LCNN-2 only adds a discriminative representation error loss to one late hidden layer. It suggests that adding direct supervision to the more category-specific late hidden layers might be more effective than to the early hidden layers which tend to be shared across categories.

ImageNet Dataset In this section, we demonstrate that LCNN can be combined with state-of-the-art CNN architecture GoogLeNet [5], which is a most recent very deep CNN with 22 layers and achieved the best performance on ILSVRC 2014. The ILSVRC classification challenge contains about 1.2 million training images and $50,000$ images for validation from 1,000 categories.

To tackle such a very deep network architecture, we add explicit supervision

Network Architecture	Top-1 (%)	Top-5 (%)
GoogLeNet [5]	-	89.93
GoogLeNet* [5] (baseline)	62.64	84.96
AlexNet [107]	58.9	-
Clarifai [107]	62.4	-
LCNN-2 (argmax)	65.38	87.09

Table 4.5: Recognition Performances using different approaches on the ImageNet 2012 Validation set. The result of GoogLeNet [5] is copied from original paper while GoogLeNet* [5] is the reproduced result by using the same parameters provided in [5]. The result of LCNN-2 is obtained by training the network from scratch under the same training condition as GoogLeNet* [5].

to multiple late hidden layers instead of a single one. Specifically, as shown in Figure 4.3(c), the discriminative representation error losses are added to three layers: loss_1/fc, loss_2/fc and $\text{Pool}_5/7\times7\text{S}_1$ with the same weights used for the three softmax loss layers in [5]. We evaluate our approach in terms of top-1 and top-5 accuracy rate. We train the LCNN-2 and GoogLeNet on the ImageNet dataset from scratch under the same training condition, and we adopt the argmax classification scheme.

The results are displayed in Table 4.5, where LCNN-2 achieved better results than GoogLeNet in both evaluation metrics under the same training condition. Please note that we did not get the same result reported GoogLeNet [5] using their parameters in the paper. Our goal here is to show that as the network becomes deeper, learning good discriminative features for hidden layers might become more difficult solely depending on the prediction error loss. Therefore, adding explicit supervision to late hidden layers under this scenario becomes particularly useful.

Caltech101 Dataset Caltech101 contains $9,146$ images from 101 object categories and a background category. In this experiment, we test the performance of LCNN with a limited amount of training data, and compare it with several state-of-the-art approaches, including those which are not deep learning based.

For fair comparison with previous work, we follow the standard classification settings. During training time, 30 images are randomly chosen from each category to form the training set, and at most 50 images per category are tested. We use the ImageNet trained model from AlexNet in [9] and VGGNet-16 in [8], and fine-tune them under our LCNN architecture. The hidden layer supervision is added to the second fully-connected layer (fc_7). We set the hyperparameter $\alpha = 0.0375$.

The performance of directly fine-tuning AlexNet and VGGNet-16, as well as finetuning LCNN with different training/classification schemes are displayed in Table 4.6. With only a limited amount of data available, our approach makes better use of the training data and achieves higher accuracy. LCNN outperforms both the deep learning approaches and other non-deep learning methods, representing state-of-the-art on this task.

Method	Accurary(%)
LC-KSVD [95]	73.6
Zeiler [92]	86.5
Dosovitskiy [108]	85.5
Zhou [109]	87.2
He [110]	91.44
AlexNet [9] (baseline)	87.1
LCNN-1 (k-NN)	88.51
LCNN-2 (argmax)	90.11
LCNN-2 (k-NN)	89.45
VGGNet-16 [8] (baseline)	92.5
LCNN-2* (argmax)	93.7
LCNN-2* (k-NN)	93.6

Table 4.6: Comparisons of LCNN with other approaches on the Caltech101 dataset. The results of LCNN-2* are obtained by using VGGNet-16 as the underlying architecture while the results of LCNN-1 and LCNN-2 are based on AlexNet.

Chapter 5: Learning a Discriminative Filter Bank within a CNN for Fine-grained Recognition

5.1 Background and Motivation

Fine-grained object recognition involves distinguishing sub-categories of the same super-category (*e.g.*, birds [34], cars [36] and aircrafts [79]), and solutions often utilize information from localized regions to capture subtle differences. Early applications of deep learning to this task built traditional multistage frameworks upon convolutional neural network (CNN) features; more recent CNN-based approaches are usually trained end-to-end and can be roughly divided into two categories: *localization-classification sub-networks* and *end-to-end feature encoding*.

Previous multistage frameworks utilize low-level CNN features to find discriminative regions or semantic parts, and construct a mid-level representation out of them for classification [60, 111–115]. These methods achieve better performance compared to two types of baselines: *(i)* they outperform their counterparts with hand-crafted features (*e.g.*, SIFT) by a huge margin, which means that low-level CNN features are far more effective than previous hand-crafted ones; *(ii)* they significantly outperform their baselines which finetune the same CNN used for feature

extraction. This further suggests that CNN's ability to learn mid-level representations is limited and still has sufficient room to improve. Based on these observations, end-to-end frameworks aim to *enhance the mid-level representation learning capability of CNN*.

The first category, *localization-classification sub-networks*, consists of a classification network assisted by a localization network. The mid-level learning capability of the classification network is enhanced by the localization information (*e.g.*part locations or segmentation masks) provided by the localization network. Earlier works from this category [116–120] depend on additional semantic part annotations, while more recent ones [121–123] only require category labels. Regardless of annotations, the common motivation behind these approaches is to first *find the corresponding parts* and then *compare their appearance*. The first step requires the semantic parts (*e.g.*head and body of birds) to be shared across object classes, encouraging the representations of the parts to be similar; but, in order to be discriminative, the latter encourages the part representations to be different across classes. This subtle conflict implies a trade-off between recognition and localization ability, which might reduce a single integrated network's classification performance. Such a trade-off is also reflected in practice, in that training usually involves alternating optimization of the two networks or separately training the two followed by joint tuning. Alternating or multistage strategies complicate the tuning of the integrated network.

The second category, *end-to-end feature encoding* [59,124–127], enhances CNN mid-level learning by encoding higher order statistics of convolutional feature maps. The need for end-to-end modeling of higher order statistics became evident when

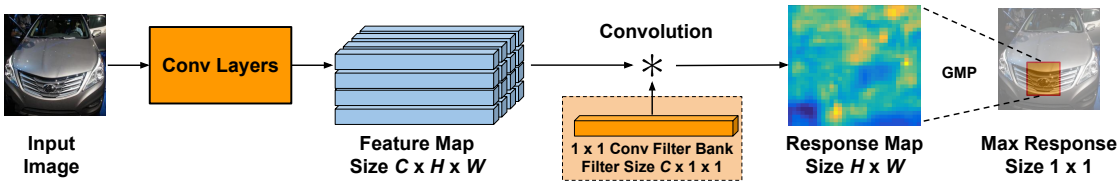

Figure 5.1: The motivation of our approach is to regard a $C \times 1 \times 1$ vector in a feature map as the representation of a small patch and a 1×1 convolutional filter as a discriminative patch detector. A discriminative patch can be discovered by convolving the feature map with the 1×1 filter and performing Global Max Pooling (GMP) over the response map. The full architecture is illustrated in Figure 5.2.

the Fisher Vector encodings of SIFT features outperformed a finetuned AlexNet by a large margin on fine-grained recognition [78]. The resulting architectures have become standard benchmarks in the literature. While effective, end-to-end encoding networks are less human-interpretable and less consistent in their performance across non-rigid and rigid visual domains, compared to localization-classification sub-networks.

This paper addresses the issues facing both categories of end-to-end networks. Our main contribution is to explicitly learn discriminative mid-level patches *within* a CNN framework in an end-to-end fashion without extra part or bounding box annotations. This is achieved by regarding 1×1 filters as small "patch detectors", designing an asymmetric multi-stream structure to utilize both patch-level information and global appearance, and introducing filter supervision with non-random layer initialization to activate the filters on discriminative patches. Conceptually, our discriminative patches differ from the parts in localization-recognition sub-networks, such that they are not necessarily shared across classes as long as they have discriminative appearance. Therefore, our network fully focuses on classification and avoids

the trade-off between recognition and localization. Technically, a convolutional filter trained as a discriminative patch detector will only yield a high response at a certain region for one class.

The resulting framework enhances the mid-level learning capability of the classical CNN by introducing a bank of discriminative filters. In practice, our framework preserves the advantages of both categories of previous approaches:

- Simple and effective. The network is easy to build and once initialized only involves single-stage training. It outperforms state-of-the-art.
- High human interpretability. This is shown through various ablation studies and visualizations of learned discriminative patches.
- Consistent performance across different fine-grained visual domains and various network architectures.

5.2 Related Work

Fine-grained recognition Research in fine-grained recognition has shifted from multistage frameworks based on hand-crafted features [41, 46, 54, 57, 78] to multi-stage framework with CNN features [60, 111, 112, 114, 115], and then to end-to-end approaches. Localization-classification sub-networks [116–119, 121, 122, 128] have a localization network which is usually a variant of R-CNN [77, 129], FCN (Fully Convolutional Network) [130] or STN (Spatial Transformer Network) [121] and a recognition network that performs recognition based on localization. More recent advances explicitly regress the location/scale of the parts using a recurrent local-

ization network such as LSTM [128] or a specifically designed recurrent architecture [122]. End-to-end encoding approaches [59, 124–127] encode higher order information. The classical benchmark, Bilinear-CNN [59] uses a symmetric two-stream network architecture and a bilinear module that computes the outer product over the outputs of the two streams to capture the second-order information. [124] further observed that similar performance can be achieved by taking the outer product over a single-stream output and itself. More recent advances reduce high feature dimensionality [124, 125] or extract higher order information with kernelized modules [126, 127]. Others have explored directions such as utilizing hierarchical label structures [131], combining visual and textual information [132–134], 3D-assisted recognition [36, 47, 135], introducing humans in the loop [37, 39, 136], and collecting larger amount of data [58, 137–139].

Intermediate representations in CNN Layer visualization [92] has shown that the intermediate layers of a CNN learn human-interpretable patterns from edges and corners to parts and objects. Regarding the discriminativeness of such patterns, there are two hypotheses. The first is that some neurons in these layers behave as "grandmother cells" which only fire at certain categories, and the second is that the neurons forms a distributed code where the firing pattern of a single neuron is not distinctive and the discriminativeness is distributed among all the neurons. As empirically observed by [140], a classical CNN learns a combination of "grandmother cells" and a distributed code. This observation is further supported by [11], which found that by taking proper weighted average over all the feature maps produced by a convolutional layer, one can effectively visualize all the regions in the input image

used for classification. Note that both [140] and [11] are based on the original CNN structure and the quality of representation learning remains the same or slightly worse for the sake of better localization. On the other hand, [103,141,142] learn more discriminative representations by putting supervision on intermediate layers, usually by transforming the fully-connected layer output through another fully-connected layer followed by a loss layer. These transformations introduce a separation between the supervisory signal and internal filters that makes their methods difficult to visualize. A more recent related work is the popular SSD [143] detection framework; it associates a convolutional filter with either a particular category of certain aspect ratio or certain location coordinates. Compared to SSD, our architecture operates at a finer-level (small patches instead of objects) and is optimized for recognition.

5.3 Learning Discriminative Patch Detectors as a Bank of Convolutional Filters

We regard a 1×1 convolutional filter as a small patch detector. Specifically, referring to Figure 5.1, if we pass an input image through a series of convolutional and pooling layers to obtain a feature map of size $C \times H \times W$, each $C \times 1 \times 1$ vector across channels at fixed spatial location represents a small patch at a corresponding location in the original image. Suppose we have learned a 1×1 filter which has high response to a certain discriminative region; by convolving the feature map with this filter we obtain a heatmap. Therefore, a discriminative patch can be found simply by picking the location with the maximum value in the entire heatmap. This operation

of spatially pooling the entire feature map into a single value is defined as Global Max Pooling (GMP) [11].

Two requirements are needed to make the feature map suitable for this idea. First, since the discriminative regions in fine-grained categories are usually highly localized, we need a relatively small receptive field, *i.e.*, each $C \times 1 \times 1$ vector represents a relatively small patch in the original image. Second, since fine-grained recognition involves accurate patch localization, the stride in the original image between adjacent patches should also be small. In early network architectures, the size and stride of the convolutional filters and pooling kernels were large. As a result, the receptive field of a single neuron in later convolutional layers was large, as was the stride between adjacent fields. Fortunately, the evolution of network architectures [5, 8, 144] has led to smaller filter sizes and pooling kernels. For example, in a 16-layer VGG network (VGG-16), the output of the 10^{th} convolutional layer `conv4_3` represents patches as small as 92×92 with stride 8, which is small and dense enough for our task given common CNN input size.

In the rest of Section 5.3, we demonstrate how a set of discriminative patch detectors can be learned as a 1×1 convolutional layer in a network specifically designed for this task. An overview of our framework is displayed in Figure 5.2. There are three key components in our design: an asymmetric two-stream structure to learn discriminative patches as well as global features (Section 5.3.1), convolutional filter supervision to ensure the discriminativeness of the patch detectors (Section 5.3.2) and non-random layer initialization to accelerate the network convergence (Section 5.3.3). We then extend our framework to handle patches of different scales (Section

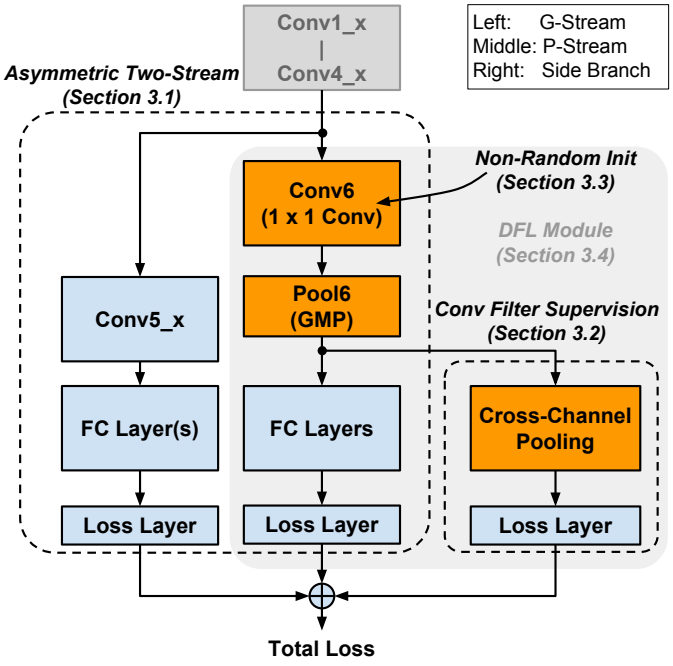

Figure 5.2: Overview of our framework, which consists of a) an asymmetric two-stream architecture to learn both the discriminative patches and global features, b) supervision imposed to learn discriminative patch detectors and c) non-random layer initialization. For simplicity, except GMP, all pooling and ReLU layers between convolutional layers are not displayed.

5.3.4). We use VGG-16 for illustration, but our ideas are not limited to any specific network architecture as our experiments show.

5.3.1 Asymmetric Two-stream Architecture

The core component of the network responsible for discriminative patch learning is a 1×1 convolutional layer followed by a GMP layer, as displayed in Figure 5.1. This component followed by a classifier (*e.g.*, fully-connected layers and a softmax layer) forms the discriminative patch stream (P-Stream) of our network, where the prediction is made by inspecting the responses of the discriminative patch detectors. The P-Stream uses the output of `conv4_3` and the minimum receptive field in this

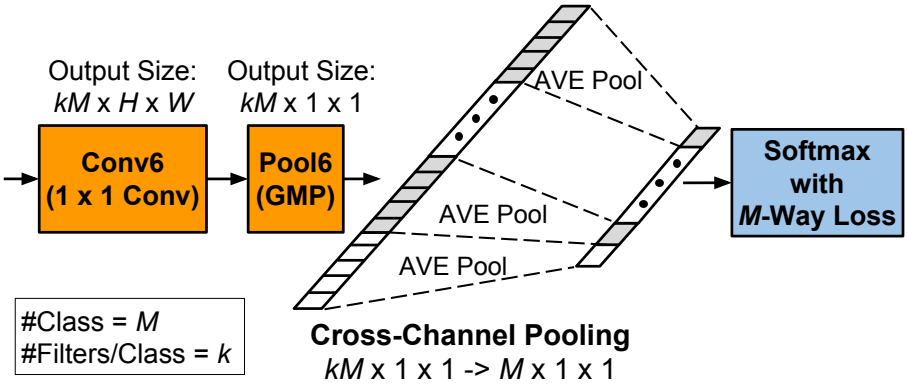

Figure 5.3: The illustration of our convolutional filter supervision. The filters in `conv6` are grouped into M groups, where M is the number of classes. The maximum responses in group i are averaged into a single score indicating the effect of the discriminative patches in Class i. The pooled vector is fed into a softmax loss layer to encourage discriminative patch learning.

feature map corresponds to a patch of size 92×92 with stride 8.

The recognition of some fine-grained categories might also depend on global shape and appearance, so another stream preserves the further convolutional layers and fully connected layers, where the neurons in the first fully connected layer encode global information by linearly combining the whole convolutional feature maps. Since this stream focuses on global features, we refer to it as the G-Stream.

We merge the two streams in the end.

5.3.2 Convolutional Filter Supervision

Using the network architecture described above, the 1×1 convolutional layer in the P-Stream is not guaranteed to fire at discriminative patches as desired. For the framework to learn class-specific discriminative patch detectors, we impose supervision directly at the 1×1 filters by introducing a Cross-Channel Pooling layer

followed by a softmax loss layer, shown in Figure 5.3 as part of the whole framework (the side branch) in Figure 5.2.

Filter supervision works as follows. Suppose we have M classes and each class has k discriminative patch detectors; then the number of 1×1 filters required is kM. After obtaining the max response of each filter through GMP, we get a kM-dimensional vector. Cross-Channel Pooling averages the values across every group of k dimensions as the response of a certain class, resulting in an M-dimensional vector. By feeding the pooled vector into an M-way softmax loss, we encourage the filters from any class to find discriminative patches from training samples of that class, such that their averaged filter response is large. We use average instead of max pooling to encourage all the filters from a given class to have balanced responses. Average pooling tends to affect all pooled filters during back propogation, while max pooling only affects the filter with the maximum response. Similar considerations are discussed in [11].

Since there is no learnable parameter between the softmax loss and the 1×1 convolutional layer, we directly adjust the filter weights via the loss function. In contrast, previous approaches which introduce intermediate supervision [103, 141, 142] have learnable weights (usually a fully-connected layer) between the side loss and the main network, which learn the weights of a classifier unused at test time. The main network is only affected by back-propogating the gradients of these weights. We believe this is a key difference of our approach from previous ones.

5.3.3 Layer Initialization

In practice, if the 1×1 convolutional layer is initialized randomly, with filter supervision it may converge to bad local minima. For example, the output vector of the Cross-Channel Pooling can approach all-zero or some constant to reduce the side loss during training, a degenerate solution. To overcome the issue, we introduce a method for non-random initialization.

The non-random initialization is motivated by our interpretation of a 1×1 filter as a patch detector. The patch detector of Class i is initialized by patch representations from the samples in that class, using weak supervion without part annotations. Concretely, a patch is represented by a $C \times 1 \times 1$ vector at corresponding spatial location of the feature map. We extract the `conv4_3` features from the ImageNet pretrained model and compute the energy at each spatial location (l_2 norm of each C-dimensional vector in a feature map). As shown in the first row of Figure 5.10, though not perfect, the heatmap of energy distribution acts as a reasonable indicator of useful patches. Then the vectors with high l_2 norms are selected via non-maximum suppression with small overlap threshold; k-means is performed over the selected C-dimensional vectors within Class i and the cluster centers are used as the initializations for filters from Class i. To increase their discriminativeness, we further whiten the initializations using [68] and do l_2 normalization. In practice this simple method provides reasonable initializations which are further refined during end-to-end training. Also, in Section 5.4 we show that the energy distribution becomes much more discriminative after training.

As long as the layer is properly initialized, the whole network can be trained in an end-to-end fashion just once, which is more efficient compared with the multistage training strategy of previous works [117–119].

5.3.4 Extension: Multiple Scales

Putting Section 5.3.1 to 5.3.3 together, the resulting framework can utilize discriminative patches from a single scale. A natural and necessary extension is to utilize patches from multiple scales, since in visual domains such as birds and aircrafts, objects might have larger scale variations.

As discussed in Section 5.3.1, discriminative patch size depends on the receptive field of the input feature map. Therefore, multi-scale extension of our approach is equivalent to utilizing multiple feature maps. We regard the P-Stream and side branch (with non-random initialization) together as a "Discriminative Filter Learning" (DFL) module that is added after `conv4_3` in Figure 5.2. By simply adding the DFL modules after multiple convolutional layers we achieve multi-scale patch learning. In practice, feature maps produced by very early convolutional layers are not suitable for class-specific operations since they carry information that is too low-level, therefore the DFL modules are added after several late convolutional layers in Section 5.4.

Our multi-layer branch-out is inspired by recent approaches in object detection [143,145], where feature maps from multiple convolutional layers are directly used to detect objects of multiple scales. Compared with these works, our approach operates

at a finer level and is optimized for recognition instead of localization.

5.4 Experiments

In the rest of this paper, we denote our approach by DFL-CNN, which is an abbreviation for *Discriminative Filter Learning within a CNN*. We use the following datasets:

CUB-200-2011 [34] has 11,788 images from 200 classes officially split into 5,994 training and 5,794 test images.

Stanford Cars [36] has 16,185 images from 196 classes officially split into 8,144 training and 8,041 test images.

FGVC-Aircraft [79] has 10,000 images from 100 classes officially split into 6,667 training and 3,333 test images.

5.4.1 Implementation Details

We first describe the basic settings of our DFL-CNN and then we introduce two higher-capacity settings. The input size of all our networks is 448×448, which is standard in the literature. We do not use part or bounding box (BBox) annotations and compare our method with other weakly-supervised approaches (without part annotation). In addition, no model ensemble is used in our experiments.

The network structure of our basic DFL-CNN is based on 16-layer VGGNet [8] and the DFL module is added after `conv4_3`, as illustrated exactly in Figure 5.2. In `conv6`, we set the number of filters per class to be 10. During Cross-Channel average

pooling, the maximum responses of each group of 10 filters are pooled into one dimension. At initialization time, `conv6` is initialized in the way discussed in Section 5.3.3; other original VGG-16 layers are initialized from an ImageNet pretrained model directly (compared with "indirect" initialization of `conv6`) and other newly introduced layers are randomly initialized. After initialization, a single stage end-to-end training proceeds, with the G-Stream, P-Stream and side branch having their own softmax with cross-entropy losses with weights 1.0, 1.0 and 0.1 respectively. At test time, these softmax-with-loss layers are removed and the prediction is the weighted combination of the outputs of the three streams.

We extend DFL-CNN in two ways. The first extension, 2-scale DFL-CNN, was discussed in Section 5.3.4. In practice, two DFL modules are added after `conv4_3` and `conv5_2`, while the output of the last convolutional layer (`conv5_3`) is used by G-Stream to extract global information. The second extension shows that our approach applies to other network architectures, a 50-layer ResNet [144] in this case. Similar to VGGNet, ResNet also groups convolutional layers into five groups and our DFL module is added to the output of the fourth group (*i.e.*`conv4_x` in [144]). Initialization, training and testing of the two extended networks are the same as basic DFL-CNN.

5.4.2 Results

The results on CUB-200-2011, Stanford Cars and FGVC-Aircraft are displayed in Table 5.1, Table 5.2 and Table 5.3, respectively. In each table from top to bottom,

the methods are separated into five groups, as discussed in Section 5.1, which are (1) fine-tuned baselines, (2) CNN features + multi-stage frameworks, (3) localization-classification subnets, (4) end-to-end feature encoding and (5) DFL-CNN. The basic DFL-CNN, 2-scale extension and ResNet extension in Section 5.4.1 are denoted by "DFL-CNN (1-scale) / VGG-16", "DFL-CNN (2-scale) / VGG-16" and "DFL-CNN (1-scale) / ResNet-50", respectively. Our VGG-16 based approach not only outperforms corresponding fine-tuned baseline by a large margin, but also achieves or outperforms state-of-the-art under the same base model; our best results further outperform state-of-the-art by a noticeable margin on all datasets, suggesting its effectiveness.

Earlier multi-stage frameworks built upon CNN features achieve comparable results, while they often require bounding box annotations and the multi-stage nature limits their potential. The end-to-end feature encoding methods have very high performance on birds, while their advantages diminish when dealing with rigid objects. The localization-classification subnets achieve high performance on various datasets, usually with a large number of network parameters. For instance, the STN [121] consists of an Inception localization network followed by four Inception classification networks without weight-sharing, and RA-CNN [122] consists of three independent VGGNets and two localization sub-networks. Our end-to-end approach achieves state-of-the-art with no extra annotation, enjoys consistent performance on both rigid and non-rigid objects, and has relatively compact network architecture.

Our approach can be applied to various network architectures. Most previous approaches in fine-grained recognition have based their network on VGGNets and

Method	Base Model	Accuracy (%)
FT VGGNet [122]	VGG-19	77.8
FT ResNet	ResNet-50	84.1
CoSeg(+BBox) [60]	VGG-19	82.6
PDFS [114]	VGGNet	84.5
STN [121]	Inception [146]	84.1
RA-CNN [122]	VGG-19	85.3
MA-CNN [123]	VGG-19	**86.5**
B-CNN [59]	VGG-16	84.1
Compact B-CNN [124]	VGG-16	84.0
Low-rank B-CNN [125]	VGG-16	84.2
Kernel-Activation [127]	VGG-16	85.3
Kernel-Pooling [126]	VGG-16	86.2
Kernel-Pooling [126]	ResNet-50	84.7
DFL-CNN (1-scale)	VGG-16	85.8
DFL-CNN (2-scale)	VGG-16	86.7
DFL-CNN (1-scale)	ResNet-50	**87.4**

Table 5.1: Comparison of our approach (DFL-CNN) to recent results on CUB-200-2011, **without** extra annotations (if not specified). For the finetuned (FT) baselines, we cite the best previously reported result if it is better than our implementation. The black-bold number represents the best previous result.

Method	Base Model	Accuracy (%)
FT VGGNet [122]	VGG-19	84.9
FT ResNet	ResNet-50	91.7
BoT(+BBox) [115]	VGG-16	92.5
CoSeg(+BBox) [60]	VGG-19	92.8
RA-CNN [122]	VGG-19	92.5
MA-CNN [123]	VGG-19	**92.8**
B-CNN [59]	VGG-16	91.3
Low-Rank B-CNN [125]	VGG-16	90.9
Kernel-Activation [127]	VGG-16	91.7
Kernel-Pooling [126]	VGG-16	92.4
Kernel-Pooling [126]	ResNet-50	91.1
DFL-CNN (1-scale)	VGG-16	93.3
DFL-CNN (2-scale)	VGG-16	**93.8**
DFL-CNN (1-scale)	ResNet-50	93.1

Table 5.2: Comparison of our approach (DFL-CNN) to recent results on Stanford Cars **without** extra annotations (if not specified).

Method	Base Model	Accuracy (%)
FT VGGNet	VGG-19	84.8
FT ResNet	ResNet-50	88.5
MGD(+BBox) [113]	VGG-19	86.6
BoT(+BBox) [115]	VGG-16	88.4
RA-CNN [122]	VGG-19	88.2
MA-CNN [123]	VGG-19	**89.9**
B-CNN [59]	VGG-16	84.1
Low-Rank B-CNN [125]	VGG-16	87.3
Kernel-Activation [127]	VGG-16	88.3
Kernel-Pooling [126]	VGG-16	86.9
Kernel-Pooling [126]	ResNet-50	85.7
DFL-CNN (1-scale)	VGG-16	91.1
DFL-CNN (2-scale)	VGG-16	**92.0**
DFL-CNN (1-scale)	ResNet-50	91.7

Table 5.3: Comparison of our approach (DFL-CNN) to recent results on FGVC-Aircraft **without** extra annotation (if not specified).

Settings	Accuracy (%)
G-Stream Only	80.3
P-Stream Only	82.0
G + P	84.9
G + P + Side	85.8

Table 5.4: Contribution of the streams *at test time* on CUB-200-2011. Note that at training time a *full* DFL-CNN model is trained, but the prediction only uses certain stream(s).

`pool6` Method	Accuracy (%)
GMP	85.8
GAP	80.4

Table 5.5: Effect of Global Max Pooling (GMP) vs. Global Average Pooling (GAP) on CUB-200-2011.

previously reported ResNet-based results are less effective than VGG-based ones. Table 5.1, 5.2 and 5.3 shows that our ResNet baseline is already very strong, however our ResNet based DFL-CNN is able to outperform the strong baseline by a large margin (*e.g.*3.3% absolute percentage on birds). This clearly indicates that CNN's mid-level learning capability can still be improved even though the network is very deep.

5.4.3 Ablation Studies

We conduct ablation studies to understand the components of our approach. These experiments use the basic DFL-CNN framework and the CUB-200-2011 dataset.

Contribution of each stream Given a trained DFL-CNN, we investigate the contribution of each stream at test time. Table 5.4 shows that the performance of the G-Stream or P-Stream alone is mediocre, but the combination of the two is significantly better than either one alone, indicating that the global information and the discriminative patch information are highly complementary. Additionally, the side branch provides extra gain to reach the full performance in Table 5.1.

Layer Initialization	Filter Supervision	Accuracy (%)
-	-	82.2
✓	-	84.4
✓	✓	85.8

Table 5.6: Effect of intermediate supervision of DFL-CNN *at training time*, evaluated on CUB-200-2011.

Method	Without BBox (%)	With BBox (%)
FT VGG-16 [131]	74.5	79.8
DFL-CNN	85.8	85.7

Table 5.7: Effect of BBox evaluated on CUB-200-2011.

Effect of intermediate supervision We investigate the effect of Section 5.3.2 and 5.3.3 by training the DFL-CNN without certain component(s) and comparing with the full model. Table 5.6 shows a significant performance improvement when we gradually add the intermediate supervision components to improve the quality of learned discriminative filters. Note that Table 5.6 does not include "Filter Supervision without Layer Initialization" settings since it leads to failure to converge of P-Stream as mentioned in Section 5.3.3.

GMP vs. GAP More insight into the training process can be obtained by simply switching the pooling method of `pool6` in Figure 5.2. As can be seen from the Table 5.5, switching the pooling method from GMP to Global Average Pooling (GAP) leads to a significant performance drop such that the accuracy is close to "G-Stream Only" in Table 5.4. Therefore, although `conv6` is initialized to the same

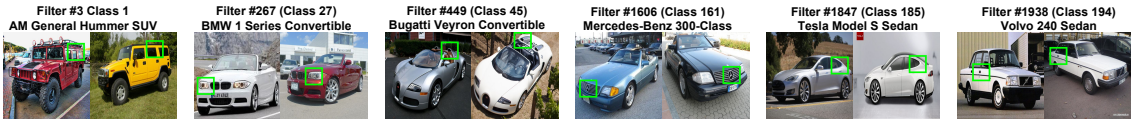

Figure 5.4: The visualization of top patches in Stanford Cars. We remap the spatial location of the highest activation in a feature map back to the patch in the original image. The results are highly consistent with human perception, and cover diverse regions such as head light (2^{nd} **column**), air intake (3^{th} **column**), frontal face (4^{th} **column**) and the black side stripe (**last column**).

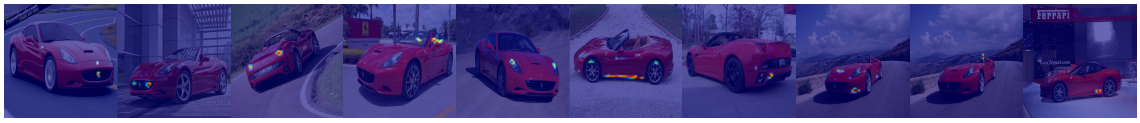

Figure 5.5: Sample visualization of all ten filter activations learned for one class (Class 102) by upsampling the `conv6` feature maps to image resolution, similar to [11]. The activations are disriminatively concentrated and cover diverse regions. Better viewed at 600%.

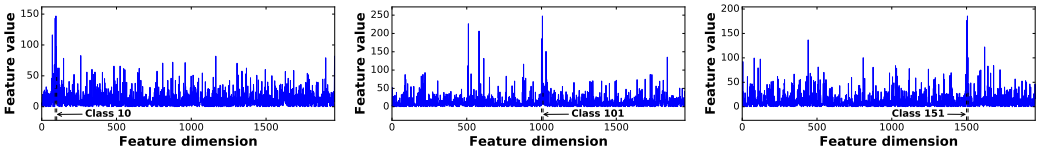

Figure 5.6: The `pool6` features averaged over all test samples from Class 10, 101 and 151 in Stanford Cars. The dash lines indicate the range of values given by the discriminative patch detectors belonging to the class. The representations peak at the corresponding class.

state, during training GMP makes the filters more discriminative by encouraging the 1×1 filters to have very high response at a certain location of the feature map and the gradients will only be back-propagated to that location, while GAP makes the P-Stream almost useless by encouraging the filters to have mediocre responses over the whole feature maps and the gradients affect every spatial location.

Unnecessary BBox. Since our approach, DFL-CNN, is able to utilize discriminative patches without localization, it is expected to be less sensitive to BBox than the fine-tuned baseline, as supported by the results in Table 5.7.

5.4.4 Visualization and Analysis

Insights into the behavior of our approach can be obtained by visualizing the effects of `conv6`, the 1×1 convolutional layer. To understand its behavior, we

- visualize patch activations. Since we regard each filter as a discriminative patch detector, we identify the learned patches by remapping spatial locations

Figure 5.7: Visualization of a failure case, where the filter activates on commonly appeared licence plates.

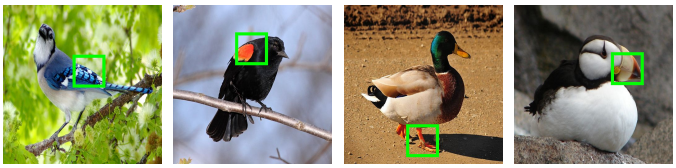

Figure 5.8: The visualization of patches in CUB-200-2011. We accurately localize discriminative patches without part annotations, such as the bright texture (**first image**), the color spot (**second image**), the webbing and beak (**third** and **forth** image).

of top filter activations back to images. Figure 5.4 shows that we do find high-quality discriminative regions.

- visualize a forward pass. Since the max responses of these filters are directly used for classification, by visualizing the output of `conv6`'s next layer, `pool6`, we find that it produces discriminative representations which have high responses for certain classes.
- visualize back propagation. During training, `conv6` can affect its previous layer, `conv4_3` (VGG-16), through back propagation. By comparing the `conv4_3` features before and after training, we find that the spatial energy distributions of previous feature maps are changed in a discriminative fashion.

5.4.4.1 Stanford Cars

The visualization of top patches found by some classes' 1×1 filters is displayed in Figure 5.4; the visualization of all ten filters learned for a sample class is displayed

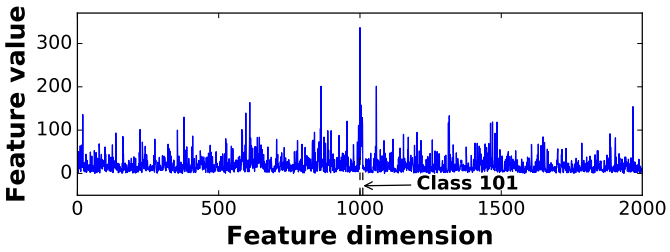

Figure 5.9: The averaged `pool6` features over all test samples from Class 101 in CUB-200-2011, peaky at corresponding dimensions.

in Figure 5.5. Unlike previous filter visualizations, which pick human interpretable results randomly among the filter activations, we have imposed supervision on `conv6` filters and can identify their corresponding classes. Figure 5.4 shows that the top patches are very consistent with human perception. For instance, the 1847^{th} filter belonging to Class 185 (Tesla Model S) captures the distinctive tail of this type. Figure 5.5 shows that the filter activation are highly concentrated at discriminative regions and the ten filters cover diverse regions. The network can localize these subtle discriminative regions because: a) 1×1 filters correspond to small patch detectors in original image, b) the filter supervision, and c) the use of cluster centers as initialization promotes diversity.

The visualization of `pool6` features is shown in Figure 5.6. We plot the averaged representations over all test samples from a certain class. Since we have learned a set of discriminative filters, the representations should have high responses at one class or only a few classes. Figure 5.6 shows that our approach works as expected. As noticeable, the fine-grained similarity at patch-level (*e.g.*Audi A4 and Audi A6) and few common patterns (example shown in Figure 5.7) might explain the alternative peaks in Figure 5.6.

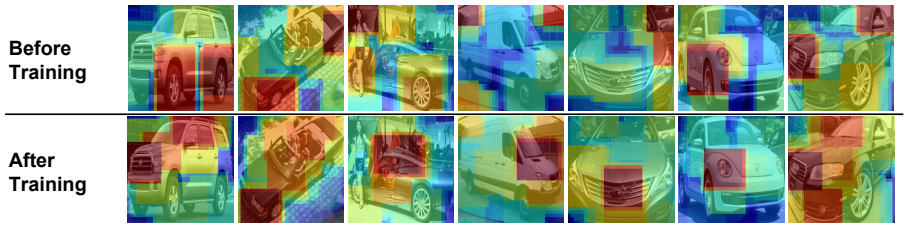

Figure 5.10: Visualization of the energy distribution of `conv4_3` feature map before and after training for Stanford Cars. We remap each spatial location in the feature map back to the patch in the original image. After training in our approach, the energy distribution becomes more discriminative. For example, in the 1st **column**, the high energy region shifts from the wheels to discriminative regions like the frontal face and the top of the vehicle; in the 2nd **column**, after training the energy over the brick patterns is reduced; in the 3rd **column**, the person no longer lies in high energy region after training; in the 7th **column**, before training the energy is focused mostly at the air grill, and training adds the discriminative fog light into the high energy region. More examples are interpretated in Section 5.4.4.1.

Most interesting is the effect of `conv6` on the previous convolutional layer `conv4_3` through back propagation. As discussed in Section 5.3.3, we use the energy distribution of `conv4_3` as a hint to provide layer initialization. After training, we observed that the energy distribution is refined by `conv6` and becomes more discriminative, as shown by Figure 5.10. We map every spatial location in the feature map back to the corresponding patch in the original image, and the value of each pixel is determined by the max energy patch covering that pixel. From the first line of Figure 5.10, the features extracted from an ImageNet pretrained model tend to have high energy at round patterns such as wheels, some unrelated background shape, a person in the image and some texture patterns, which are common patterns in generic models found in [92]. After training, the energy shifts from these patterns to discriminative regions of cars. For example, in the 6th column, the feature map has high energy initially at both the wheel and the head light; after training, the network

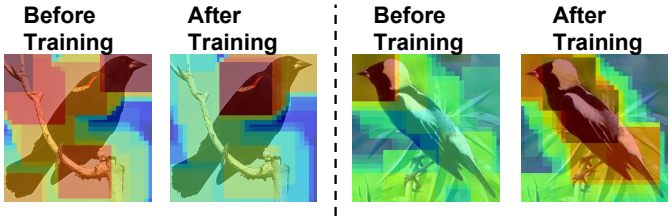

Figure 5.11: The energy distributions of `conv4_3` feature maps before and after training in CUB-200-2011. After training, in the left example, the high energy region at the background branches is greatly shrinked and the energy is concentrated at the discriminative color spot; in the right example, more energy is distributed to the distinctive black-and-white wing and tail of the species.

has determined that a discriminative patch for that class (Volkswagen Beetle) is the head light rather than the wheels. Therefore, `conv6` have beneficial effects on their previous layer during training.

5.4.4.2 CUB-200-2011

Figure 5.8 shows examples of the discriminative patches found by our approach. They include the texture and spots with bright color as well as specific shape of beak or webbing. Compared with visualizations of previous works not using part annotations (*e.g.* [59, 60]), our approach localizes such patches more accurately because our patch detectors operate over denser and smaller patches and do not have to be shared across categories.

Similar to cars, features from the next GMP layers are peaky at certain categories (Fig. 5.9). The energy distributions of previous convolutional features are also improved: high energy at background regions like branches is reduced and the discriminative regions become more focused or diverse according to different categories (Fig. 5.11).

Chapter 6: Continuous 3D Pose Estimation for Fine-grained Objects

6.1 Motivation

Estimating 3D object pose from a single 2D image is an indispensable step in various practical applications, such as fine-grained object recognition [147, 148], car damage detection [149], novel view synthesis [150, 151], grasp planning [152] and autonomous driving [153].

The human visual system has a remarkable ability of interpreting 3D shapes and structures [154]. Even with a single image, human are still able to predict the 3D pose and the 3D shape of objects [155]. With the recent development of deep Convolutional Neural Networks (CNNs) [156], 3D object pose estimation in

Table 6.1: Comparison of our dataset with some of the other 3D datasets.

	# class	# image	# instance	annotation type	fine grained
3D Object	10	6,675	✗	discretized view	✗
EPFL Cars	1	2,299	✗	continuous view	✗
Pascal 3D+	12	30,899	79	2d-3d alignment	✗
ObjectNet3D	100	90,127	201,888	2d-3d alignment	✗
StanfordCars (Ours)	196	16,185	16,185	2d-3d alignment	✓
FGVC-Aircraft (Ours)	100	10,000	10,000	2d-3d alignment	✓
CompCars (Ours)	113	5,696	5696	2d-3d alignment	✓
Total (Ours)	409	31881	31881	2d-3d alignment	✓

static images has become a new challenging problem in computer vision [157]. Most existing work utilize CNNs to directly map the images into the 3D pose space by regressing the 6 degrees of freedom (DoF) of objects [157–159]. By simultaneously estimating 3D pose and class label, with a ground truth label-shape correspondence dataset that could generate 3D shape prior from a pure label [160], the CNNs are able to recover most of the 3D information for rigid objects such as cars, airplanes, etc [161]. This type of methods is very straightforward and works pretty well, particularly when there are enough training data with 3D annotations [162].

However, due to the expensive annotation cost, most existing 3D pose estimation datasets only provide ground truth pose for a few object classes and the number of instances associated to each category is still quite small [163]. This could prevent the CNNs from learning robust models for 3D pose estimation. To the best of our knowledge, there are so far only two large scale 3D dataset, Pascal 3D+ [161] and ObjectNet3D [162]. Both dataset are designed for the 3D pose estimation for general objects, and there is still no large scale 3D pose dataset for fine-grained objects. In this work, we introduce a new dataset that is able to benchmark both fine-grained object recognition and fine-grained object pose estimation. Specifically, we augment the existing Stanford Cars [147], FGVC-Aircraft [164] and CompCars [165] dataset with the annotation of ground truth 3D pose for each instance, producing a total number of 30000+ images and 300+ classes, with approximately 100 images per category. Table 1 shows the general statistics of the 3D pose dataset.

Our dataset annotation process is similar to ObjectNet3D [162]. We first download a set of computer aided design (CAD) models from ShapeNet [160]. These

CAD models are selected to match the different object categories in each fine-grained dataset. Then each object instance in a category is associated with the corresponding CAD model. We then ask our mechanical turkers to align the 3D geometry to best match a 2D image using our designed interface. It is non-trivial to align the 2D image with a 3D model, since different pose parameters may create a similar visual appearance, making the annotators easily confused. To guarantee the quality of the 3D pose alignment, we designed a user friendly annotation tool to allow the annotators to easily visualize the quality of the pose matching and hence able to adjust to a better pose with ease. In the end, we obtain the aligned 3D poses for all fine-grained objects in our selected 2D images across the three fine-grained dataset. Figure 1 shows our annotation interface and some examples in our dataset.

The new annotation allows us to study the problem of joint continuous fine-grained 3D pose estimation and fine-grained object recognition [159]. To address this new problem, We introduce a new approach and a new 3D shape representation. In terms of the approach, we augment the recently introduced Mask R-CNN [166] which proves to be successful on object detection and semantic segmentation, and add an extra branch to conduct 3D pose estimation. In terms of the shape representation, we introduce *location field*, a new representation that models the 3D shape with location values of all the pixels on the object surface. This new representation is very efficient in representing the whole 3D shape, particularly it allows CNNs efficiently learn useful filters that can capture the global and local shape of the objects.

Our contribution is three-fold. First, we collect a new large 3D pose dataset

for fine-grained objects, which consists of more than 30,000 images with more than 300 different categories. Second, we propose a new multi-task network structure for simultaneous fine-grained recognition and 3D pose estimation. Third, we propose *location field*, a new 3D representation that efficiently encodes the object shapes. We conduct exhaustive experiments on our new dataset. Experimental results suggest our method achieves new state-of-the-art performance on simultaneous fine-grained recognition and pose estimation.

6.2 Related Work

Our work is joint fine-grained recognition and 3D pose estimation in 2D images. We first briefly review fine-grained recognition, then focus on reviewing 3D pose estimation.

6.2.1 Fine-Grained Recognition

Fine-grained recognition refers to the task of distinguishing sub-ordinate categories, such as bird species [167], dog breeds [168], car models [147], airplane categories [164], etc. Great success has been achieved for fine-grained recognition in the last few years, thanks to the collection of large scale fine-grained dataset [165, 169–171]. In addition, the methodology for fine-grained recognition is also evolving fast [172, 173]. Based on the recent success of CNNs, many variants such as part models [174] and attention models [175, 176] are developed to capture the level of need for fine-grained recognition. While most existing models only base on monoc-

ular images [170, 173, 177], the 3D shape is found to be a very important additive cue for recognizing fine-grained objects [147, 178]. However, almost all existing fine-grained dataset are lack of 3D pose labels and 3D shape information. In this work, we fix this gap by annotating ground truth 3D poses on three different popular fine-grained dataset including Stanford Cars [147], FGVC-Aircraft [164] and CompCars [165]. This facilitates our study on (1) how much 3D shape can potentially improve fine-grained recognition, and (2) how much joint fine-grained recognition and pose estimation can help each other.

6.2.2 Monocular 3D Pose Estimation

Monocular 3D pose estimation is a very challenging problem in computer vision. Since a monocular image does not contain direct depth information, many previous approaches attempt to address this problem through keypoint matching or template matching methods. With the advance of deep learning, the 2D based 3D object pose estimation methods in the literature can be roughly clustered into three main groups, (1) feature / keypoint based methods (2) template / appearance based methods, and (3) deep learning based methods.

Feature / keypoint based methods: The feature-based methods first extract local discriminative features from points of interests [179], then match them to features on the 3D models to establish the 2D-3D keypoint correspondence [180–185]. The matching result is usually represented by a (probabilistic) heatmap of 2D keypoints. After then, the 3D pose is obtained by finding the best alignment from

the detected 2D keypoints to the 3D keypoints. In more details, Collet *et al.* [186] propose an iterative framework that iteratively generates groups of image features that are likely to belong to a single object instance and computes object hypotheses given clusters of features Ramakrishna *et al.* [187] recover the 3D configuration of a human pose from 2D locations of the human anatomical landmarks in a single image. Ghodrati *et al.* [188] consider viewpoint estimation as a 1-vs-all classification problem and show that template based methods with fisher encoding or CNN encoding can outperforms keypoint based methods. Zhou *et al.* [189] represent 3D shape as a linear combination of rotatable basis shapes to address the simultaneous camera parameter estimation and 3D shape estimation. Feature-based methods are able to handle occlusions between objects. However, they require sufficient textures on the objects in order to extract discriminative features for keypoint matching. To deal with texture-less objects, [180, 190] propose to learn feature descriptors automatically.

Template / appearance based methods: Template based methods are widely used in 3D pose estimation [191–198]. By representing a 3D object with a set of independent 2D appearance models, with each model for one viewpoint, the methods try to directly figure out the most probable pose by finding the best template match in the input image. Compared to the keypoint based methods, template based methods are more useful in detecting pose for texture-less objects. For example, Branchman *et al.* [194] show a single RGB image is sufficient to achieve visually convincing results for 3D pose estimation. Cao *et al.* [195] obtains real-time 3D pose estimation for textureless objects by matching the real images with a

3D model to render example poses of a textureless object. All these methods use hand-engineered features such as HOG and fisher encoding which is not end-to-end trainable.

Deep learning based methods: With the recent development of deep learning, many new approaches start to adopt CNNs for 3D pose estimation [190, 199–206]. The overall idea of most methods are relatively simple, instead of designing various complicated matching methods or feature engineering, most of them learn CNNs that maps the 2D image directly to the 3D pose space. For example, Doumanoglou *et al.* [190] use CNNs to estimate 3D pose through directly regressing object poses by exploiting Siamese Networks. Doumanoglou *et al.* [201] further provide an end-to-end neural architecture for simultaneous object detection and 3D pose estimation. Yang *et al.* [207] propose an auto-masking neural networks that automatically learn to select the most discriminative object parts across different viewpoints from training images. Poirson *et al.* [158, 208] extends the SSD model for object detection to pose estimation and show the SSD is also good at modeling pose. More recently, Xiang *et al.* [157] carefully design a convolutional neural network, PoseCNN, specifically for 3D pose estimation, and achieve state-of-the-art results on the challenging dataset with severe object occlusion. Our method also lies in this category, where we propose to use the Mask R-CNN as the baseline neural network structure to conduct pose estimation.

6.2.3 Monocular 3D Pose Estimation Dataset

Following with the methods, there are several monocular 3D pose benchmark dataset for studying pose estimation [161, 162, 209–215]. For example, 3D Object dataset provides viewpoint annotation for 10 object classes with 10 instances for each class . EPFL Car dataset consists of 2,299 images of 20 car instances at multiple azimuth angles. To avoid the variance in other pose parameters, the elevation and distance in this dataset is almost the same for all the car instances. The KITTI dataset [216] provides 3D bounding box annotation for two categories (car and pedestrian), where there are 80,000 instances for each category. However, there is no detailed shape or classes for each object. The IKEA Object dataset [210] provides dense 3D annotations of 800 images for 90 different IKEA objects. Their dataset is limited to indoor images and the number of instances per category is small. The NYC3DCars dataset [211] annotate 5,186 car images along with 567K 3D points but only the dataset contains only one category. Pascal 3D+ is the first large scale 3D pose dataset for generic objects, with 12 different object categories and 30,899 images from the challenging VOC Pascal dataset [217]. The most recent ObjectNet3D dataset [162] further builds a large scale database for 3d object recognition, that consists of 100 categories and 90,127 images.

Our goal is to provide a new large scale dataset to enhance fine-grained object recognition and 3D pose estimation on a challenging and real world benchmark. Compared to other large generic object datasets such as Pascal 3D+ [197] and ObjectNet3D [162], we annotate a large group of fined-grained objects. For example,

we annotate the 3D pose for a total number of 200 different car categories and 100 different airplane categories. To the best of our knowledge, this is the first fine-grained object recognition dataset with 3D pose annotation. Table 1 shows a comparison between our dataset and some of the most relevant datasets mentioned above.

6.2.4 Joint Object Recognition and 3D Pose Estimation

3D pose estimation often comes with joint object recognition since both problems are fundamental and can potentially help each other. There are several works attempting to address joint object recognition and 3D pose estimation in monocular images [159, 180, 183, 186, 202, 218–220]. There are extensive works addressing the problem of joint object detection and 3D pose estimation [157, 158, 191–193, 197, 200, 207, 208, 213, 221–225].

Savarese and Fei-Fei [218] first propose to solve the problem of joint object recognition and 3D pose estimation, where they provide a 3D part representation to model the object appearance around the surface. Hao *et al.* [221] follow this work and propose a dense, multiview representation for the 3D objects parameterized by a triangular mesh of viewpoints. Later, [191] and [197] extend the discriminative deformable part based models to achieve joint object detection and viewpoint estimation. With the new success of deep learning, Elhoseiny *et al.* [202] adjust a pre-trained CNN model to work on the problem of joint object recognition and pose estimation, and [200, 222, 226] all attempt to solve the problem of joint object

detection and pose estimation through end-to-end deep CNNs. In all these works, however, they restrice the pose to only either the azimuth or yaw angle, not the full 3D pose information. Mahendran *et al.* [159] propose to use CNNs to recover the full 3D rotation matrix which is a much harder problem. We also attempt to recover the full 3D pose information, which is similar to [159]. But we focus on the problem of fine-grained object recognition jointly with 3D pose estimation, and no previous methods have studied along this direction.

6.2.5 3D Representation

Various works [227–229] study what is an appropriate presentation for 3D shapes. Since there is still no conclusion on what 3D representation is the best for 3D object recognition and pose estimation, we propose

6.3 Dataset

6.3.1 3D models

We build three fine-grained 3D pose dataset, two for vehicles and one for airplanes. The dataset consist of two parts, i.e., 2D images and 3D models. The 2D images vehicles are collected from StanfordCars [147] and CompCars [165] and the 2D images of the airplane dataset are collected from FGVC-Aircraft [164]. Target objects in all 2D images of these three dataset are non-occluded and easy to identify. all 2D images from the dataset. In order to distinguish the difference between fine-grained categories, we adopt a distinct model for each fine-grained class.

Thanks to the work [160], large numbers of 3D models for fine-grained vehicles and airplanes are available with make/model names in their meta data. 3D models used in our work are courtesy from ShapeNet [160]. We find the 3D model for an image category by searching the category name in ShapeNet meta data and pick the matched model. The matched model is very accurate since it usually is from the same make/model. For StanfordCars and FGVC-Aircraft, we include images from all 196 and 100 categories. For CompCars, we include 113 categories with matched 3D models in ShapeNet. Note that, our dataset is the very first one which employs fine-grained category aware 3D model in 3D pose estimation.

6.3.2 Camera model

The world coordinate system is defined in accordance with the 3D model coordinate system. In this case, a point P in a 3D model is projected onto a point p on a 2D image:

$$p = \mathcal{P}P \tag{6.1}$$

where $\mathcal{P}$ is a projection matrix:

$$\mathcal{P} = K\,[R|T] \tag{6.2}$$

K is the intrinsic parameter:

$$K = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

R is a 3×3 rotation matrix encoding the rotation transformation between the world coordinate system and the camera coordinate system. This transformation is given by three angels i.e., elevation, azimuth and in plane rotation. We assume that the camera is always facing to the origin of the 3D model. Hence the translation $T = [0, 0, d]^t$ is only defined up to model depth d, the distance between the origins of two coordinate systems, and the principal point (u, v) is the projection of the origin of world coordinate system on the image. As a result, our model has 7 parameters in total: camera focal length: f, principal point location x_c, y_c, azimuth a, elevation e, in-plane rotation θ and model depth d.

Compared with the camera models which are used in [161, 162] where 6 parameters are to be estimate, our camera model formulates both the camera focal length and object depth at the same time for the sake of a more realistic model which achieves a better fitting results than models having 6 parameters.

6.3.3 3D Annotation

We annotate 3D pose information for all 2D images in our three dataset through crowd-source. To facilitate the annotation process, we developed an an-

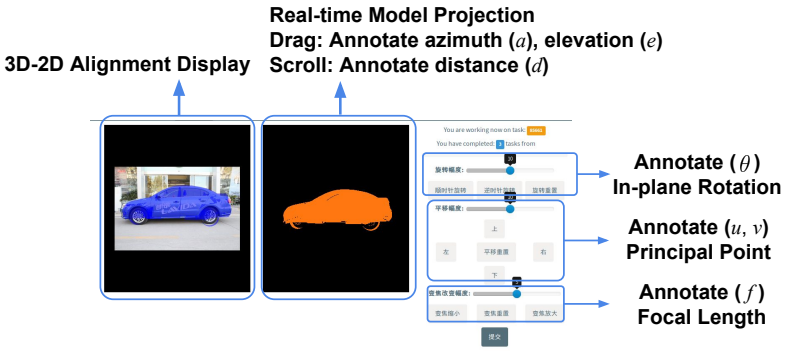

Figure 6.1: Overview of our annotation interface.

notation tools as illustrated in Figure 6.1. For each image under annotation, a 3D model is chosen according to the fine-grained car type which is given beforehand. Then, an annotator is asked to adjust the seven parameters so that the projected 3D model is aligned with the target object in 2D image. This process can be roughly summarized as follows: (1) shift the 3D model such that the center of the model (the origin of the world coordinate system) is roughly aligned with the center of the target object in the 2D image; (2) rotating the model to the same orientation as the target object in 2D image; (3) adjusting d and f to match the size of the target object in 2D image. Some finer adjustment might be applied after the three main steps.

6.4 Continuous 3D Pose Estimation for Fine-Grained Objects

Given an input image of a fine-grained object, our task is to predict *all* the parameters accurately in Equation 6.4, *i.e.*, 3D rotation R, distance d and intrinsic parameters (u, v) and f such that the projected 3D model aligns with the object in the image as well as possible.

The major advantage of our datasets over previous ones is that we have more accurate 3D models corresponding to each fine-grained category. These newly introduced correspondences can possibly add new supervisory signals at training time. Moreover, we introduced a dense 3D representation called "3D location field"; complementary to standard "sparse" 3D representation of pose parameters, we explore its usage in supervising 3D pose estimation.

All our 3D pose estimation frameworks are based on recent 2-stage proposal-based detection architectures (*i.e.*, Faster R-CNN and Mask R-CNN). Inspired by recent success of Mask R-CNN, we further explore the new problem of joint fine-grained recognition and 3D pose estimation in an end-to-end fashion. As discussed in the introduction, simultaneously providing the fine-grained label and accurate 3D pose in a single network is potentially useful in several real-world applications.

6.4.1 Baseline Framework

Our baseline method only uses 2D appearance information to regress pose parameters, and only 2D images are needed as input at test time. The baseline is derived from Faster R-CNN [230]. Casting our pose estimation problem into a detection framework is naturally motivated by the dependency on 2D appearance and the relation between the two tasks. Since we are not using key points as an intermediate representation or as an attention mechanism, performing pose estimation within the region of interest (RoI) gets rid of unrelated image regions and makes the usage of 2D information much more effective. In addition, the estimation of 3D

pose, especially the intrinsic parameters in Equation 6.3, is highly correlated with the detection task. For instance, the principal point (u, v) (the projection of 3D object center on 2D image) is highly related to the center of the RoI; f, the parameter controlling the scale of the projection, is related to the diameter of the RoI. In this way, pose estimation benefits from additional supervisory signals provided by detection.

The choice of the regression targets is of vital importance to achieve good performance in practice. Therefore we carefully modify the parametrization of 3D poses from that at annotation time.

- The rotation parameters R is parametrized by azimuth a, elevation e and in-plane rotation θ during annotation. During training, directly regressing the angles might not be a good idea since the angles are periodic and a small difference in input appearance can lead to a large loss (*e.g.*, 359° and 1° in azimuth). To avoid this problem, we parametrize 3D rotation using the *quaternion* representation, which can be regressed from 2D appearance alone. Although less human-interpretable than angles, regressing quaternion achieves better performance since small difference in appearance corresponds to small difference in quaternions.

- 3D Distance d can also be regressed from 2D appearance alone. Since a larger d means that the object in the image will have more obvious perspective distortion.

- (u, v) is highly related to RoI center. Therefore, we regress $(\Delta u, \Delta v)$, the

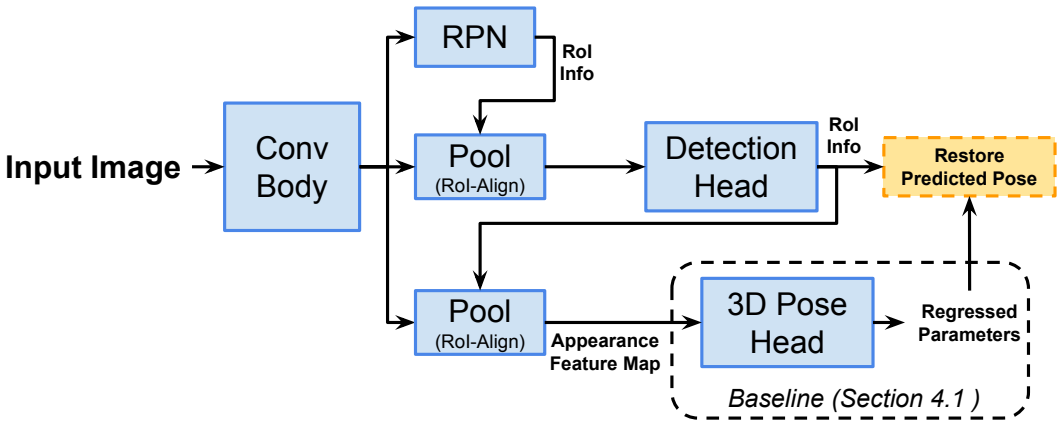

Figure 6.2: Pose Estimation Framework Diagram

offset of the principal point from the RoI center. The offset exists and can be regressed from 2D appearance, since the projection of the 3D object center might not necessarily be the 2D center depending on the poses.

- f is related to the diameter of the RoI. Therefore, we regress the ratio $\hat{f}$ between the original f and the diameter defined as the square root of RoI area. Such ratio needs regression since the relationship between f and diameter is nonlinear and depends on the poses.

The modification of network architecture is relatively straightforward. As shown in Figure 6.2, we add a pose estimation branch along with the existing class prediction and bounding box regression branches. Similar to the bounding box regression branch, the estimation of each group of pose parameters consists of a fully-connected (FC) layer and a smooth l_1 loss. The centers and sizes of RoIs are also used to adjust the regression targets at training time and generate the final predictions at test time, as discussed above.

6.4.2 Improve Pose Estimation via 3D Location Field

Given an object in an image and its 3D model, a 3D location field maps every foreground pixel to its corresponding location on the surface of the 3D model, *i.e.*, $f(x, y) = (X, Y, Z)$. The resulting field has the same size of the image and has three channels containing the X, Y and Z coordinates respectively. Two sample images of car and aircraft and their corresponding 3D location fields can be seen in Figure 6.3. Different from previous works which use a sparse collection of 3D coordinates, the 3D location field is a dense representation of 3D information from which the underlying 3D pose can be inferred.

A 3D location field can be easily generated from the 3D model and our 3D annotation in three steps. In Step 1 the 3D location of the virtual camera center *w.r.t.* the 3D model center is calculated as

$$C = [d\cos(e)\cos(a),\ d\cos(e)\sin(a),\ d\sin(e)], \tag{6.4}$$

where a, e, d denote azimuth, elevation and distance in Section 6.3, respectively. In Step 2, given a 3D model containing a set of faces (usually triangular faces) and known camera center C, all the visible faces can be obtained using the Z-buffer algorithm. In Step 3, the visible faces are projected to the image plane and the mapping between the 2D pixels and the 3D coordinates are established.

Our first extension of baseline is to improve 3D pose estimation using 3D location field, based on Mask R-CNN. We still expect only 2D image input at test

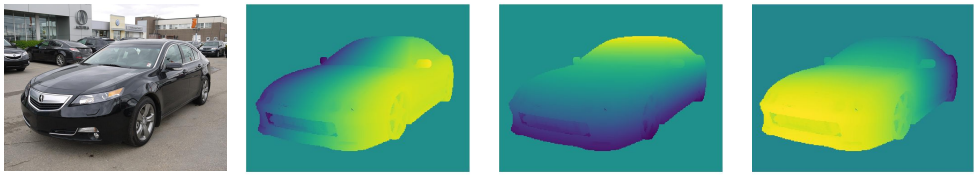

Figure 6.3: Sample image and its corresponding 3D location fields

time, therefore we regress 3D location field and use the regressed field to help pose estimation. We modify the mask branch of Mask R-CNN to regress 3D location field, and the regressed field along with previous RoI feature map are used by the 3D pose branch. The regressed location field passes convolution layer followed by pooling layer and merges with the pooled appearance feature map. After that comes the FC layers and smooth l_1 losses to regress the pose parameters as discussed in Section 6.4.1. The detailed network architecture is shown in Figure 6.4.

The field is very suitable for the task due to the following reasons: (i) the convolution layer can be easily applied to extract features since the field preserves 2D locality; (ii) the field only encodes 3D location information without any rendering of 3D model and naturally avoids the domain gap between synthetic image and photo-realistic images; (iii) the field is invariant to color, texture and scale of the images.

6.4.3 Joint Pose Estimation and Fine-grained Recognition

Our second extension is to tackle the more challenging problem of joint fine-grained recognition and 3D pose estimation in an end-to-end fashion. Given an input 2D image at test time, a single network will produce both the fine-grained label and 3D pose parameters.

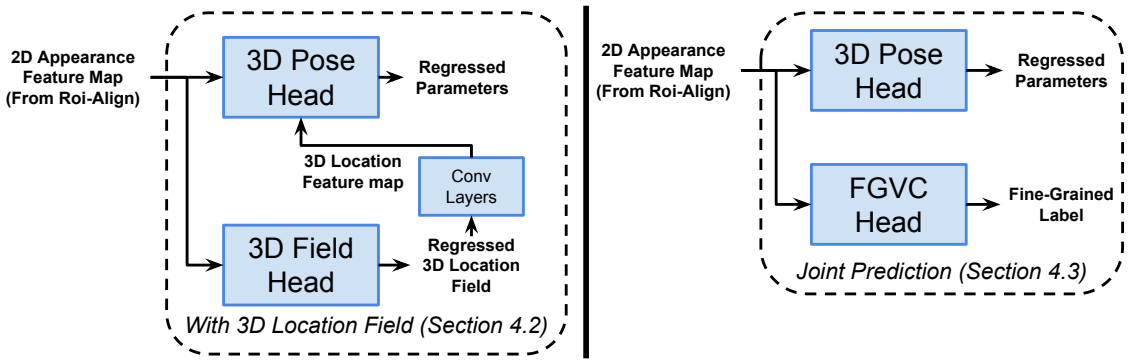

Figure 6.4: Left: Network architecture of using 3D location field to help pose. Right: Network architecture for joint fine-grained recognition and pose estimation.

We add a fine-grained recognition branch to our baseline framework in Section 6.4.1. In Mask R-CNN and our baseline, all the branches share all the convolution layer weights. In our joint prediction architecture, the fine-grained recognition branch does not share weights of its late convolution layers with the detection branch and 3D pose branch, as illustrated in Figure 6.4. The reason is that the detection and pose branch encourages the representations of different fine-grained categories to be similar while the fine-grained recognition branch needs them to be discriminative. In practice, this early branch-out strategy proves important to achieve good performance.

6.5 Baseline Experiments

6.5.1 Evaluation Metrics

There are various metrics to evaluate 3D pose estimation depending on different parametrization. Our focus is to evaluate the overall quality of perspective projection rather than any particular parameter. Therefore our evaluation met-

ric is based on **Average Distance of Model Points** in [231], which measures the averaged distance between predicted projected points and their corresponding ground truth projections. According to [231], this is the most widely-used pose error function.

Concretely, given one test sample $\mathcal{S} = \{\hat{\mathcal{P}}, \mathcal{P}, \mathcal{M}\}$, where its predicted pose is $\hat{\mathcal{P}}$, its ground truth pose $\mathcal{P}$ and corresponding 3D model $\mathcal{M}$, the Average Distance of Model Points is defined as

$$e_{\text{ADD}}(\mathcal{S}) = \underset{\mathbf{X} \in \mathcal{M}}{\text{avg}} \left\| \mathcal{P}\mathbf{X} - \hat{\mathcal{P}}\mathbf{X} \right\|_2 \tag{6.5}$$

The unit of the above distance is number of pixels. To make the metric scale-invariant, we normalize it using the diameter of the bounding box. We denote the normalized distance as $\tilde{e}_{\text{ADD}}$. To measure the performance over the whole test set, we compute mean and median of $\tilde{e}_{\text{ADD}}$ over all test samples. Also, by setting threshold on $\tilde{e}_{\text{ADD}}$, we can get an accuracy number Acc_{th}. In practice, the common threshold is 0.1, which means that the prediction with average projection error less than 10% of the diameter is considered correct.

It is worth mentioning that the 3D models are only used when computing the evaluation metrics. During test time, only a single 2D image is fed into the network to predict the pose $\mathcal{P}$.

Name	Stanford Cars 3D	FGVC-Aircraft 3D	CompCars 3D
# Train	8144	6667	3798
# Test	8041	3333	1898

Table 6.2: Train / Test Split of the Datasets.

6.5.2 Experimental Settings

Data Split: For Stanford Cars 3D and FGVC-Aircraft 3D, since we annotated all images, we follow the standard train / test split provided by the original dataset provider [147] [79]. For CompCars 3D, we randomly sample 2/3 of our annotated data as training set and the rest 1/3 as test set. The number of training and test samples of the three datasets are displayed in Table 6.2.

Implementation Details: Our implementation is based on the Detectron package [232], which includes Faster / Mask R-CNN implementations. The convolution body (*i.e.*, the "backbone" in [166]) used for the baseline is ResNet-50. For fair comparison, the convolution body is initialized from ImageNet pre-trained model, and other layers are randomly initialized (*i.e.*, we are not using COCO pre-trained detectors). Following the settings of Mask R-CNN, the whole network is trained end-to-end; at test time, a cascaded strategy is adopted, where the 3D pose branch is applied only to the highest scoring box predictions after non-maximum suppression.

	Median $\tilde{e}_{\text{ADD}}$	Mean $\tilde{e}_{\text{ADD}}$	$Acc_{\text{th}=0.1}$ (%)
Stanford Cars 3D	0.1096	0.1884	45.96
FGVC-Aircraft 3D	0.0988	0.1399	50.87
CompCars 3D	0.1275	0.1580	32.52
CompCars 3D (FT)	0.0878	0.1123	58.58

Table 6.3: Baseline results of 3D pose estimation.

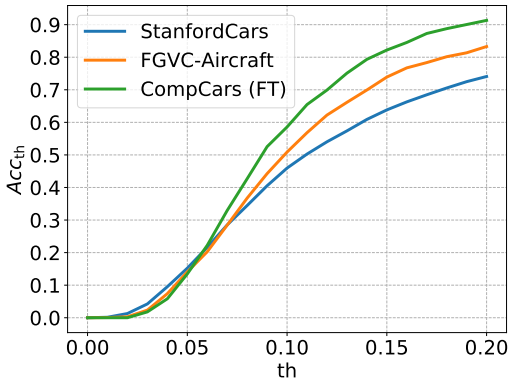

Figure 6.5: Plot of Acc_{th} *w.r.t.* threshold.

6.5.3 Results and Analysis

The baseline results for the three datasets are shown in Table 6.3. The changes of Acc_{th} *w.r.t* the threshold for the three datasets are shown in Figure 6.5. For Stanford Cars 3D and FGVC-Aircraft 3D, the *Median Average Distance of Points* (Median $\tilde{e}_{\text{ADD}}$) is around 0.1, meaning that the average projection discrepancy is less than 10% of the diameter for around half of the test samples ($Acc_{\text{th}=0.1}$). In terms of *Mean Average Distance of Points* (Mean $\tilde{e}_{\text{ADD}}$), Stanford Cars 3D has noticeably larger error than FGVC-Aircraft 3D. The main reason is that the photos of aircrafts are usually taken from a distance, which have less perspective distortion than photos of cars. For CompCars 3D dataset, besides ImageNet initialization we also report

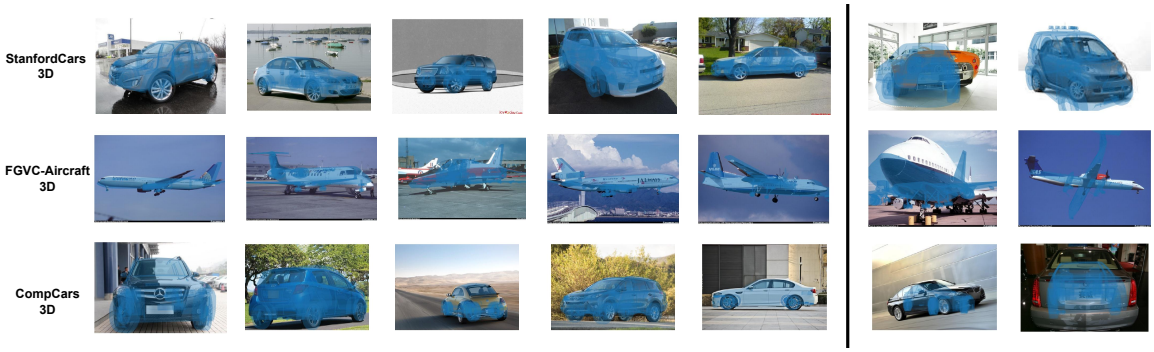

Figure 6.6: Visualizations of predicted poses for test samples. For each dataset, we show five examples of successful predictions and two of the failure cases, separated by the solid black line in the figure.

the result finetuned from a Stanford Cars 3D pretrained model, since the number of training samples is relatively small. From the last two rows of Table 6.3 we can see the effectiveness of transfer learning from Stanford Cars 3D to CompCars 3D.

We visualize the predicted poses for all three datasets in Figure 6.6. As shown by the left part of Figure 6.6, our method is able to handle poses of various orientations, scales and locations of the projection. On the other hand, as shown by the right part of Figure 6.6, failure cases exists in our predictions including some severe ones. There are rooms of improvement especially for the estimation of scale, cases with large perspective distortion and some uncommon poses with few training samples.

The promising baseline results suggest that we can accurately recover the full perspective projection from a single image.

Method	Backbone	Median $\tilde{e}_{\text{ADD}}$	Mean $\tilde{e}_{\text{ADD}}$	$Acc_{\text{th=0.1}}$ (%)	FG Acc (%)
Baseline	ResNet-50	0.1096	0.1884	45.96	-
w/ Location Field	ResNet-50	0.1004	0.1219	49.74	-
Joint Prediction	ResNet-50	0.1128	0.2081	40.93	90.87
Change Backbone	ResNet-101	0.0917	0.1075	57.83	-

Table 6.4: Extended experimental results on Stanford Cars 3D

6.6 Extended Experiments

In this section we demonstrate our extended experiments on Stanford Cars 3D dataset, including the usage of 3D location field (Section 6.5.2), joint fine-grained recognition and pose estimation (Section 6.5.3) and change of convolution body.

Effect of Location Field: The experimental results on Stanford Cars 3D with/without 3D location field is shown in the first two rows of Table 6.4. Note that the field only participates in training and at test time the network input is still a single image and the field is regressed. As can be seen, adding 3D location fields improves all the metrics of pose estimation. The reason why it is more helpful in reducing large pose error (Mean $\tilde{e}_{\text{ADD}}$) might be that the regressed field (14×14) is at relatively low resolution.

Joint Prediction of 3D Poses and Fine-grained Labels: The result of joint prediction is shown in the third row of Table 6.4, showing we achieve comparable results in both tasks. As discussed in Section 6.5.3, the 3D pose branch and the fine-grained recognition branch use separate late convolution layers (*i.e.*, `res-5`, the 5th stage of ResNet). Sharing weights of these convolution layers between the two

tasks leads to severely inferior performance.

Change of Convolution Body: Lastly, we show that our architecture is applicable to different base CNN architectures by replacing the ResNet-50 backbone with ResNet-101. As shown in the last row of Table 6.4, with larger GPU memory consumption and longer training time, ResNet-101 backbone gives better results.

Chapter 7: Conclusion

Visual feature learning is at the core of various computer vision tasks, and discriminativeness plays a major role for features for visual recognition. Fine-grained recognition is a very good task to study discriminative feature learning since it requires the recognition system to capture subtle intra-class differences. The research in fine-grained recognition has shifted from multi-stage frameworks built upon CNN features, to end-to-end CNN-based frameworks, to multi-task frameworks obtaining more than category labels. We have proposed several approaches to address these related problems.

(1) Our earlier work proposed a method based on sparse/low-rank analysis which is able to capture subtle differences when the image instances are well-aligned.

(2) We proposed a mid-level patch-based approach for fine-grained recognition. We first introduce triplets of patches with two geometric constraints to improve localizing patches, and automatically mine discriminative triplets to construct mid-level representations for fine-grained recognition. Experimental results demonstrated that our discriminative triplets mining framework performs very well on both mid-scale and large-scale fine-grained recognition datasets, and outperformed or obtained comparable results to the state-of-the-art at the time.

(3) We proposed the Label Consistent Neural Network, a supervised feature learning algorithm, by adding explicit supervision to late hidden layers. By introducing a discriminative representation error and combining it with the traditional prediction error in neural networks, we achieved better classification performance at the output layer, and more discriminative representations at the hidden layers. Experimental results show that our approach operated at the state-of-the-art at the time on several publicly available action and object recognition datasets. It leads to faster convergence speed and works well when only limited video or image data is presented. Our approach can be seamlessly combined with various network architectures. Future work includes applying the discriminative learned category-specific representations to other computer vision tasks besides action and object recognition.

(4) We proposed an approach to fine-grained recognition based on learning a discriminative filter bank within a CNN framework in an end-to-end fashion without extra annotation. This is done via an asymmetric multi-stream network structure with convolutional layer supervision and non-random layer initialization. Our approach learns high-quality discriminative patches. It obtained state-of-the-art performance at the time on both rigid / non-rigid fine-grained datasets.

(5) We further study the problem of continuous 3D pose estimation for fine-grained objects, with three major contributions. First, we collect a new fine-grained 3D dataset, which consists of more than 30,000 images with more than 300 different categories. Second, we propose a new multi-task network structure for 3D pose estimation. Third, we propose location field, a new efficient representation for 3D shapes.

Bibliography

[1] G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *Proc. ICCV*, 2011.

[2] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *arXiv: 1507.02159*, 2015.

[3] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[4] Hao Ye, Zuxuan Wu, Rui-Wei Zhao, Xi Wang, Yu-Gang Jiang, and Xiangyang Xue. Evaluating two-stream CNN for video classification. In *ICMR*, 2015.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[6] Zhenzhong Lan, Ming Lin, Xuanchong Li Alexander G. Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015.

[7] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://www.thumos.info/, 2015.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[10] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[11] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[12] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 2011.

[13] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proc. ICML*, 2010.

[14] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE TPAMI*, 2013.

[15] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 1998.

[16] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[17] R. Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 2010.

[18] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 1991.

[19] X. He, D. Cai, S. Yan, and H. Zhang. Neighborhood preserving embedding. In *Proc. ICCV*, 2005.

[20] X. He and P. Niyogi. Locality preserving projections. In *Proc. NIPS*, 2003.

[21] D. Guillamet and J. Vitrià. Non-negative matrix factorization for face recognition. In *Proc. CCIA*. 2002.

[22] H. Zhang, Z. Lin, and C. Zhang. A counterexample for the validity of using nuclear norm as a convex surrogate of rank. In *Proc. ECML PKDD*. 2013.

[23] H. Zhang, Z. Lin, C. Zhang, and J. Gao. Robust latent low rank representation for subspace clustering. *To appear in Neurocomputing*, 2014.

[24] Z. Zhang, S. Yan, and M. Zhao. Similarity preserving low-rank representation for enhanced data representation and effective subspace learning. *Neural Networks*, 2014.

[25] Z. Zhang, S. Yan, and M. Zhao. Robust image representation and decomposition by laplacian regularized latent low-rank representation. In *Proc. IJCNN*, 2013.

[26] Z. Zhang, C. Liu, and M. Zhao. Handwriting representation and recognition through a sparse projection and low-rank recovery framework. In *Proc. IJCNN*, 2013.

[27] M. Yin, S. Cai, and J. Gao. Robust face recognition via double low-rank matrix recovery for feature extraction. In *Proc. ICIP*, 2013.

[28] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):711–720, 1997.

[29] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Technical report, UILU-ENG-09-2215*, 2009.

[30] J. Cai, E. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 2010.

[31] K. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE TPAMI*, 2005.

[32] T. Sim, S. Baker, and M. Bsatn. The cmu pose, illumination, and expression (pie) database. In *Proc. FG*, 2002.

[33] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Peitro Perona. Caltech-ucsd birds 200. In *Technical Report CNS-TR-2010-001, Caltech,*, 2010.

[34] Catherine Wah, Steve Branson, Peter Welinder, Peitro Perona, and Serge Belongie. The caltech-ucsd birds 200-2011 dataset. In *Technical Report CNS-TR-2011-001, Caltech,*, 2011.

[35] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

[36] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representation for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition*, 2013.

[37] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *ECCV*, 2010.

[38] Jia Deng, Jonathan Krause, and Li Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR*, 2013.

[39] Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity comparisons for interactive fine-grained categorization. In *CVPR*, 2014.

[40] Jiongxin Liu, Angjoo Kanazawa, David W. Jacobs, and Peter N. Belhumeur. Dog breed classification using part localization. In *ECCV*, 2012.

[41] Thomas Berg and Peter N. Belhumeur. POOF: part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013.

[42] Thomas Berg and Peter N. Belhumeur. How do you tell a blackbird from a crow? In *ICCV*, 2013.

[43] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*, 2014.

[44] Ryan Farrell, Om Oza, Ning Zhang, Vlad I. Morariu, Trevor Darrell, and Larry S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011.

[45] Ning Zhang, Ryan Farrell, and Trevor Darrell. Pose pooling kernels for subcategory recognition. In *CVPR*, 2012.

[46] Ning Zhang, Ryan Farrell, Forrest N. Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013.

[47] Yen-Liang Lin, Vlad I. Morariu, Winston H. Hsu, and Larry S. Davis. Jointly optimizing 3d model fitting and fine-grained classification. In *ECCV*, 2014.

[48] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like paris? *ACM Trans. Graph.*, 31(4):101, 2012.

[49] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.

[50] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[51] Mayank Juneja, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[52] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Mid-level deep pattern mining. In *CVPR*, 2015.

[53] Shulin Yang, Liefeng Bo, Jue Wang, and Linda G. Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012.

[54] Bangpeng Yao, Gary R. Bradski, and Li Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[55] Efstratios Gavves, Basura Fernando, Cees G. M. Snoek, Arnold W. M. Smeulders, and Tinne Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013.

[56] Jian Pu, Yu-Gang Jiang, Jun Wang, and Xiangyang Xue. Which looks like which: Exploring inter-class relationships in fine-grained visual categorization. In *ECCV*, 2014.

[57] Yuning Chai, Victor S. Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *ICCV*, 2013.

[58] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, 2015.

[59] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015.

[60] Jonathan Krause, Hailin Jin, Jianchao Yang, and Fei-Fei Li. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[61] Olivier Duchenne, Francis R. Bach, In-So Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. In *CVPR*, 2009.

[62] Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P^3 & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.

[63] Ron Zass and Amnon Shashua. Probabilistic graph and hypergraph matching. In *CVPR*, 2008.

[64] Alexander Freytag, Erik Rodner, and Joachim Denzler. Birds of a feather flock together - local learning of mid-level representations for fine-grained recognition. In *ECCV Workshop on Parts and Attributes*, 2014.

[65] Christoph Göring, Erik Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *CVPR*, 2014.

[66] Jonathan Krause, Timnit Gebru, Jia Deng, Li-Jia Li, and Li Fei-Fei. Learning features and parts for fine-grained recognition. In *ICPR*, 2014.

[67] Vittorio Ferrari, Tinne Tuytelaars, and Luc J. Van Gool. Wide-baseline multiple-view correspondences. In *CVPR*, 2003.

[68] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[69] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[70] Basura Fernando, Élisa Fromont, and Tinne Tuytelaars. Mining mid-level features for image classification. *International Journal of Computer Vision*, 108(3):186–203, 2014.

[71] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James J. Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3d scene understanding. In *BMVC*, 2012.

[72] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[73] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[74] Andrea Vedaldi and Brian Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, pages 1469–1472, 2010.

[75] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[77] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[78] Philippe Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014.

[79] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013.

[80] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[81] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Rich feature hierarchies for accurate object detection and semantic segmentation. In *ICLR*, 2015.

[82] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In *ICML*, 2010.

[83] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei-Fei Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[84] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[85] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[86] Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. In *BMVC*, 2015.

[87] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[88] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[89] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[90] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv: 1207.0580*, 2012.

[91] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[92] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[93] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[94] John Wright, Allen Y. Yang, Arvind Ganesh, Shankar S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *TPAMI*, 31(2):210–227, 2009.

[95] Zhuolin Jiang, Zhe Lin, and Larry S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *CVPR*, 2011.

[96] Meng Yang, Lei Zhang, Xiangchu Feng, and David Zhang. Fisher discrimination dictionary learning for sparse representation. In *ICCV*, 2011.

[97] Qiang Qiu, Zhuolin Jiang, and Rama Chellappa. Sparse dictionary-based representation and recognition of action attributes. In *ICCV*, 2011.

[98] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012.

[99] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report, 2009.

[100] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.

[101] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 675–678, 2014.

[102] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[103] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[104] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

[105] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *ICML*, 2013.

[106] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[107] Shuo Yang, Ping Luo, Chen Change Loy, Kenneth W. Shum, and Xiaoou Tang. Deep representation learning with target coding. In *AAAI*, 2015.

[108] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Bro. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014.

[109] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.

[110] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[111] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, 2015.

[112] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015.

[113] Dequan Wang, Zhiqiang Shen, Jie Shao, Wei Zhang, Xiangyang Xue, and Zheng Zhang. Multiple granularity descriptors for fine-grained categorization. In *ICCV*, 2015.

[114] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *CVPR*, 2016.

[115] Yaming Wang, Jonghyun Choi, Vlad Morariu, and Larry S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, 2016.

[116] Ning Zhang, Jeff Donahue, Ross B. Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014.

[117] Di Lin, Xiaoyong Shen, Cewu Lu, and Jiaya Jia. Deep LAC: deep localization, alignment and classification for fine-grained recognition. In *CVPR*, 2015.

[118] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *CVPR*, 2016.

[119] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, 2016.

[120] Xiu-Shen Wei, Chen-Wei Xie, and Jianxin Wu. Mask-cnn: Localizing parts and selecting descriptors for fine-grained image recognition. *CoRR*, abs/1605.06878, 2016.

[121] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[122] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017.

[123] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*, 2017.

[124] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[125] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017.

[126] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

[127] Sijia Cai, Wangmeng Zuo, and Lei Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 2017.

[128] Michael Lam, Behrooz Mahasseni, and Sinisa Todorovic. Fine-grained recognition as hsnet search for informative image parts. In *CVPR*, 2017.

[129] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[130] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[131] Feng Zhou and Yuanqing Lin. Fine-grained image classification by exploring bipartite-graph labels. In *CVPR*, 2016.

[132] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *CVPR*, 2016.

[133] Zeynep Akata, Scott E. Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015.

[134] Xiangteng He and Yuxin Peng. Fine-grained image classification via combining vision and language. In *CVPR*, 2017.

[135] Jakub Sochor, Adam Herout, and Jiri Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *CVPR*, 2016.

[136] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *CVPR*, 2016.

[137] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015.

[138] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*, 2016.

[139] Saihui Hou, Yushan Feng, and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *ICCV*, 2017.

[140] Pulkit Agrawal, Ross B. Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.

[141] Zhuolin Jiang, Yaming Wang, Larry S. Davis, Walt Andrews, and Viktor Rozgic. Learning discriminative features via label consistent neural network. *CoRR*, abs/1602.01168, 2016.

[142] Xiaojie Jin, Yunpeng Chen, Jian Dong, Jiashi Feng, and Shuicheng Yan. Collaborative layer-wise discriminative learning in deep neural networks. In *ECCV*, 2016.

[143] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016.

[144] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[145] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.

[146] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[147] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshops on 3D Representation and Recognition*, 2013.

[148] Jakub Sochor, Adam Herout, and Jiri Havel. BoxCars: 3D boxes as cnn input for improved fine-grained vehicle recognition. In *CVPR*, 2016.

[149] Srimal Jayawardena et al. *Image based automatic vehicle damage detection*. PhD thesis, The Australian National University, 2013.

[150] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016.

[151] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *CVPR*, 2017.

[152] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *IROS*, 2017.

[153] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D object detection for autonomous driving. In *CVPR*, 2016.

[154] Scott O Murray, Daniel Kersten, Bruno A Olshausen, Paul Schrater, and David L Woods. Shape perception reduces activity in human primary visual cortex. *Proceedings of the National Academy of Sciences*, 99(23):15164–15169, 2002.

[155] Derek Hoiem, Alexei A Efros, and Martial Hebert. Geometric context from a single image. In *ICCV*, 2005.

[156] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[157] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[158] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *CVPR*, 2017.

[159] Siddharth Mahendran, Haider Ali, and Rene Vidal. Joint object category and 3d pose estimation from 2d images. *arXiv preprint arXiv:1711.07426*, 2017.

[160] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[161] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *WACV*, 2014.

[162] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. ObjectNet3D: A large scale database for 3D object recognition. In *ECCV*, 2016.

[163] Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009.

[164] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[165] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015.

[166] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[167] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[168] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, page 1, 2011.

[169] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[170] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320. Springer, 2016.

[171] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist challenge 2017 dataset. *arXiv preprint arXiv:1707.06642*, 2017.

[172] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[173] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015.

[174] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based R-CNNs for fine-grained category detection. In *ECCV*, 2014.

[175] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[176] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 842–850. IEEE, 2015.

[177] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[178] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James J Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3d scene understanding. *International Journal of Robotics Research*, 30(13):1543–1552, 2011.

[179] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[180] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

[181] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *CVPR*, 2015.

[182] J Krishna Murthy, GV Sai Krishna, Falak Chhaya, and K Madhava Krishna. Reconstructing vehicles from a single image: Shape priors for road scene understanding. In *ICRA*, 2017.

[183] Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Pose induction for novel object categories. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 64–72, 2015.

[184] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.

[185] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-DOF object pose from semantic keypoints. In *ICRA*, pages 2011–2018, 2017.

[186] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.

[187] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3D human pose from 2D image landmarks. In *ECCV*, 2012.

[188] Amir Ghodrati, Marco Pedersoli, and Tinne Tuytelaars. Is 2D information enough for viewpoint estimation? In *BMVC*, 2014.

[189] Xiaowei Zhou, Spyridon Leonardos, Xiaoyan Hu, Kostas Daniilidis, et al. 3D shape estimation from 2D landmarks: A convex relaxation approach. In *CVPR*, 2015.

[190] Andreas Doumanoglou, Vassileios Balntas, Rigas Kouskouridas, and Tae-Kyun Kim. Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation. *arXiv preprint arXiv:1607.02257*, 2016.

[191] Chunhui Gu and Xiaofeng Ren. Discriminative mixture-of-templates for viewpoint classification. In *ECCV*, 2010.

[192] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *CVPR*, 2012.

[193] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3D object detection: A real time scalable approach. In *ICCV*, 2013.

[194] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single rgb image. In *CVPR*, 2016.

[195] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. In *ICRA*, 2016.

[196] Michele Fenzi, Laura Leal-Taixé, Jörn Ostermann, and Tinne Tuytelaars. Continuous pose estimation with a spatial ensemble of Fisher regressors. In *ICCV*, 2015.

[197] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *NIPS*, 2012.

[198] Zorah Lahner, Emanuele Rodola, Frank R Schmidt, Michael M Bronstein, and Daniel Cremers. Efficient globally optimal 2d-to-3d deformable shape matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2185–2193, 2016.

[199] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In *ICCV*, 2015.

[200] Francisco Massa, Renaud Marlet, and Mathieu Aubry. Crafting a multi-task CNN for viewpoint estimation. *arXiv preprint arXiv:1609.03894*, 2016.

[201] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6D object pose and predicting next-best-view in the crowd. In *CVPR*, 2016.

[202] Mohamed Elhoseiny, Tarek El-Gaaly, Amr Bakry, and Ahmed Elgammal. A comparative analysis and study of multiview CNN models for joint object categorization and pose estimation. In *ICML*, 2016.

[203] Kota Hara, Raviteja Vemulapalli, and Rama Chellappa. Designing deep convolutional neural networks for continuous object orientation estimation. *arXiv preprint arXiv:1702.01499*, 2017.

[204] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *ICCV*, 2017.

[205] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. *arXiv preprint arXiv:1711.08848*, 2017.

[206] Siddharth Mahendran, Haider Ali, and René Vidal. 3D pose regression using convolutional neural networks. In *ICCV*, volume 1, page 4, 2017.

[207] Linjie Yang, Jianzhuang Liu, and Xiaoou Tang. Object detection and viewpoint estimation with auto-masking neural network. In *ECCV*, 2014.

[208] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C Berg. Fast single shot detection and pose estimation. In *3DV*, 2016.

[209] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *ICRA*, 2011.

[210] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing IKEA objects: Fine pose estimation. In *ICCV*, 2013.

[211] Kevin Matzen and Noah Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 761–768. IEEE, 2013.

[212] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014.

[213] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In *ECCV*, 2016.

[214] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *ICRA*, 2017.

[215] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6D object pose estimation. *arXiv preprint*, 2017.

[216] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[217] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[218] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *ICCV*, 2007.

[219] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1911, 2015.

[220] Roozbeh Mottaghi, Yu Xiang, and Silvio Savarese. A coarse-to-fine model for 3D pose estimation and sub-category recognition. In *CVPR*, 2015.

[221] Hao Su, Min Sun, Li Fei-Fei, and Silvio Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*, 2009.

[222] Francisco Massa, Mathieu Aubry, and Renaud Marlet. Convolutional neural networks for joint object detection and pose estimation: A comparative study. *arXiv preprint arXiv:1412.7190*, 2014.

[223] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Košecká. 3D bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[224] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, pages 924–933. IEEE, 2017.

[225] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3D object detection and pose estimation for grasping. In *ICRA*, 2014.

[226] Markus Braun, Qing Rao, Yikang Wang, and Fabian Flohr. Pose-rcnn: Joint object detection and pose estimation using 3d object proposals. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1546–1551. IEEE, 2016.

[227] M Zeeshan Zia, Michael Stark, Bernt Schiele, and Konrad Schindler. Detailed 3D representations for object recognition and modeling. *PAMI*, 35(11):2608–2623, 2013.

[228] M Zeeshan Zia, Michael Stark, and Konrad Schindler. Explicit occlusion modeling for 3D object class representations. In *CVPR*, 2013.

[229] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017.

[230] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[231] Tomas Hodan, Jiri Matas, and Stepán Obdrzálek. On evaluation of 6d object pose estimation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pages 606–619, 2016.

[232] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.